Document made available under the Patent Cooperation Treaty (PCT)

International application number: PCT/JP05/004190

International filing date: 10 March 2005 (10.03.2005)

Document type: Certified copy of priority document

Document details: Country/Office: JP
Number: 2004-075404
Filing date: 16 March 2004 (16.03.2004)

Date of receipt at the International Bureau: 12 May 2005 (12.05.2005)

Remark: Priority document submitted or transmitted to the International Bureau in compliance with Rule 17.1(a) or (b)

World Intellectual Property Organization (WIPO) - Geneva, Switzerland
Organisation Mondiale de la Propriété Intellectuelle (OMPI) - Genève, Suisse

PCT/JP2005/004190

日本国特許庁
JAPAN PATENT OFFICE

10.3.2005

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。
This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日 Date of Application:	２００４年　３月１６日
出願番号 Application Number:	特願２００４－０７５４０４
パリ条約による外国への出願に用いる優先権の主張の基礎となる出願の国コードと出願番号 The country code and number of your priority application, to be used for filing abroad under the Paris Convention, is	ＪＰ２００４－０７５４０４
出願人 Applicant(s):	株式会社ターボデータラボラトリー

２００５年　４月２０日

特許庁長官
Commissioner,
Japan Patent Office

小川洋

出証番号　出証特２００５－３０３６１３６

【書類名】　特許願
【整理番号】　PK040006
【あて先】　特許庁長官　殿
【国際特許分類】　G06F 17/30
　　G06F 15/00
【発明者】
　【住所又は居所】　神奈川県横浜市神奈川区松見町４丁目１１０１番地７　コートハウス菊名８０４号
　【氏名】　古庄　晋二
【特許出願人】
　【識別番号】　502369012
　【氏名又は名称】　株式会社ターボデータラボラトリー
【代理人】
　【識別番号】　100099715
　【弁理士】
　【氏名又は名称】　吉田　聡
【手数料の表示】
　【予納台帳番号】　231855
　【納付金額】　21,000円
【提出物件の目録】
　【物件名】　特許請求の範囲　1
　【物件名】　明細書　1
　【物件名】　図面　1
　【物件名】　要約書　1

【書類名】特許請求の範囲
【請求項１】
ツリー型データ構造を構成するノード間の親子関係を記憶装置上で表現する方法であって、
ルート・ノード以外のノードである非ルート・ノードの各々に対して、該非ルート・ノードの親ノードを関連付けることにより前記ノード間の親子関係を表現する、
方法。
【請求項２】
ツリー型データ構造を記憶装置上に構築する方法であって、
ルート・ノードを含むノードに固有のノード識別子を付与するノード定義ステップと、
前記ルート・ノード以外のノードである非ルート・ノードの各々に付与されたノード識別子に、前記非ルート・ノードの各々の親ノードに付与されたノード識別子を関連付ける親子関係定義ステップと、
を含む方法。
【請求項３】
前記ノード定義ステップは前記ノード識別子として数値を用いる、請求項２記載の方法。
【請求項４】
前記ノード定義ステップは前記ノード識別子として連続する整数を用いる、請求項２記載の方法。
【請求項５】
前記ノード間の親子関係は、前記非ルート・ノードの各々と、前記関連付けられた親ノードと、の組の配列によって表現される、請求項１記載の方法。
【請求項６】
前記親子関係定義ステップは、前記非ルート・ノードの各々に付与されたノード識別子と、前記関連付けられた親ノードに付与されたノード識別子と、の組の配列を前記記憶装置に格納する、請求項２記載の方法。
【請求項７】
ツリー型データ構造を記憶装置上に構築する方法であって、
同じ世代のノードよりも子ノードを優先して、ルート・ノードを含むノードに固有の連続する整数を付与するノード定義ステップと、
前記ルート・ノード以外のノードである非ルート・ノードの各々に付与された整数の順に、前記非ルート・ノードの各々の親ノードに付与された整数を並べることにより形成される配列を前記記憶装置に格納する親子関係定義ステップと、
を含む方法。
【請求項８】
前記ノード定義ステップは、
最初にルート・ノードに番号を付与するステップと、
既に番号が付与されたあるノードに唯一の子ノードが存在する場合には、当該子ノードに当該あるノードに付与された前記番号の次の番号を付与するステップと、
既に番号が付与されたあるノードに複数の子ノードが存在する場合には、当該複数の子ノードの間の兄弟関係に従って、弟ノードは直上の兄ノードの全ての子孫ノードに番号が付与された後に次の番号が付与されるように、一番上の兄ノードから一番下の弟ノードまで番号を付与するステップと、
を含む、請求項７記載の方法。
【請求項９】
ツリー型データ構造を記憶装置上に構築する方法であって、
子ノードよりも同じ世代のノードを優先して、ルート・ノードを含むノードに固有の連続する整数を付与するノード定義ステップと、
前記ルート・ノード以外のノードである非ルート・ノードの各々に付与された整数の順

に、前記非ルート・ノードの各々の親ノードに付与された整数を並べることにより形成される配列を前記記憶装置に格納する親子関係定義ステップと、
を含む方法。
【請求項１０】
前記ノード定義ステップは、
各ノードが前記ルート・ノードから何世代目のノードであるか、及び、各世代に含まれるノード数を算出するステップと、
最初に前記ルート・ノードに番号を付与するステップと、
ある世代に含まれる全てのノードに番号が付与されたならば、当該ある世代の次の世代にノードが存在しなくなるまで、当該次の世代に含まれる全てのノードに対して、親ノードが異なる場合には、当該親ノードに番号が付与された順番に当該ノードに番号を付与し、当該親ノードが同一である場合には、当該親ノードから派生した複数の子ノードの間に兄弟関係を定義し、一番上の兄ノードから一番下の弟ノードまで直前に付与された番号の次の番号から連続的に変化する固有の整数を順に付与するステップと、
を含む、請求項９記載の方法。
【請求項１１】
前記配列から、あるノードに付与された整数以上の値が格納されている連続領域を抽出することにより、前記あるノードの全ての子孫ノードを特定するステップを更に有する、請求項７又は８記載の方法。
【請求項１２】
前記配列から、あるノードに付与された整数と同じ値が格納されている連続領域を抽出することにより、前記あるノードの全ての子ノードを特定するステップを更に有する、請求項９又は１０記載の方法。
【請求項１３】
ツリー型データ構造を記憶装置上に構築する方法であって、
ルート・ノードから始めて全てのノードに連続的に変化する整数を一意に割り当てるステップと、
ノード間に親子関係を定義するステップと、
を含み、
前記全てのノードに整数を一意に割り当てるステップは、
同じ世代のノードよりも子ノードを優先して番号を付与する深さ優先モードと、子ノードよりも同じ世代のノードを優先して番号を付与する幅優先モードのどちらのモードでノードに番号を付与するかを選択するステップと、
前記深さ優先モードが選択された場合に、深さ優先でノードを検索し、検索された順にノードに番号を付与するステップと、
前記幅優先モードが選択された場合に、幅優先でノードを検索し、検索された順にノードに番号を付与するステップと、
を含み、
前記ノード間に親子関係を定義するステップは、子ノードに付与された番号の順に、当該子ノードに対応する親ノードに付与された番号を前記記憶装置に格納するステップを含む、
方法。
【請求項１４】
前記ノード間に親子関係を定義するステップは、
子ノードから親ノードへの関係を定義する子親表現モードと、親ノードから子ノードへの関係を定義する親子表現モードのどちらのモードで親子関係を定義するかを選択するステップと、
前記子親表現モードが選択された場合に、子ノードに付与された番号の順に、当該子ノードに対応する親ノードに付与された番号を前記記憶装置に格納するステップと、
前記親子表現モードが選択された場合に、親ノードに付与された番号の順に、当該親ノ

ードに対応する子ノードに付与された番号の配列を前記記憶装置に格納するステップと、
を含む、請求項１３記載の方法。
【請求項１５】
記憶装置上で親子関係を用いて表現されたツリー型データ構造の表現形式を変換する方法であって、
前記親子関係は、子ノードに付与された番号の順に、当該子ノードに対応する親ノードに付与された番号を前記記憶装置に格納することにより定義され、
同じ世代のノードよりも子ノードを優先してノードに番号を付与する深さ優先モードで表現されたツリー型データ構造の各ノードの世代を判定し、世代毎に属しているノードの個数を計数するステップと、
前記世代毎に属しているノードの個数に基づいて、子ノードよりも同じ世代のノードを優先してノードに番号を付与する幅優先モードで番号を付与する際に各世代において付与される番号を決定するステップと、
前記判定されたノードの世代及び前記決定された各世代において付与される番号に基づいて、前記各ノードの番号を前記幅優先モードで付与される番号に変換する変換配列を作成するステップと、
前記各ノードの前記親子関係を、前記変換配列を使用して前記幅優先モードで付与される番号で表現された親子関係に変換するステップと、
を有する方法。
【請求項１６】
記憶装置上で親子関係を用いて表現されたツリー型データ構造の表現形式を変換する方法であって、
前記親子関係は、子ノードに付与された番号の順に、当該子ノードに対応する親ノードに付与された番号を前記記憶装置に格納することにより定義され、
子ノードよりも同じ世代のノードを優先してノードに番号を付与する幅優先モードで表現されたツリー型データ構造の各ノードの子孫の数を計数するステップと、
親ノードに付与されるべき番号に、当該ノードと同一の親ノードから派生した兄弟ノードのうち当該ノードよりも前に番号が付与されている兄ノードの数及び当該兄ノードの子孫の数を加算することにより、前記幅優先モードで付与された番号を、同じ世代のノードよりも子ノードを優先してノードの番号を付与する深さ優先モードで付与される番号に変換する変換配列を作成するステップと、
前記各ノードの前記親子関係を、前記変換配列を使用して前記深さ優先モードで付与される番号で表現された親子関係に変換するステップと、
を有する方法。
【請求項１７】
記憶装置上で親子関係を用いて表現されたツリー型データ構造の表現形式を変換する方法であって、
前記親子関係は、子ノードに付与された番号の順に、当該子ノードに対応する親ノードに付与された番号を前記記憶装置に格納することにより定義され、
子ノードよりも同じ世代のノードを優先してノードに番号を付与する幅優先モードで表現されたツリー型データ構造の各ノードを深さ優先で検索し、前記幅優先モードで付与された番号を、同じ世代のノードよりも子ノードを優先してノードの番号を付与する深さ優先モードで付与される番号に変換する変換配列を作成するステップと、
前記各ノードの前記親子関係を、前記変換配列を使用して前記深さ優先モードで付与される番号で表現された親子関係に変換するステップと、
を有する方法。
【請求項１８】
記憶装置上で親子関係を用いて表現されたツリー型データ構造の表現形式を変換する方法であって、
前記親子関係は、子ノードに付与された番号の順に、当該子ノードに対応する親ノード

に付与された番号を第１の配列の要素として前記記憶装置に格納することにより定義され、

各ノードに関して、当該ノードに付与された番号が前記第１の配列の要素として出現する回数を計数するステップと、

前記各ノードに関して当該ノードに対応する子ノードに付与された番号を格納するため、前記計数された回数に応じた個数の連続領域を前記記憶領域に第２の配列として確保するステップと、

前記第１の配列の要素を順次に読み出し、当該要素と値が一致する番号が付与されたノードのために確保された前記第２の配列の要素として、前記第１の配列の要素に対する子ノードの番号を順次に格納するステップと、

を有する方法。

【請求項１９】

記憶装置上で親子関係を用いて表現されたツリー型データ構造の表現形式を変換する方法であって、

前記親子関係は、親ノードに付与された番号の順に、当該親ノードに対応する子ノードに付与された番号を第１の配列の要素として前記記憶装置に格納することにより定義され、

子ノードに付与された番号の順に、当該子ノードに対応する親ノードに付与された番号を格納するため、前記記憶装置に第２の配列を確保するステップと、

前記第１の配列の要素を順次に読み出し、当該要素と値が一致する番号が付与されたノードのために確保された前記第２の配列の要素として、前記第１の配列の要素に対する親ノードの番号を順次に格納するステップと、

を有する方法。

【請求項２０】

ツリー型データ構造を記憶装置上に構築する情報処理装置であって、

ルート・ノードを含むノードに固有のノード識別子を付与するノード定義手段と、

前記ルート・ノード以外のノードである非ルート・ノードの各々に付与されたノード識別子に、前記非ルート・ノードの各々の親ノードに付与されたノード識別子を関連付ける親子関係定義手段と、

を含む情報処理装置。

【請求項２１】

前記ノード定義手段は前記ノード識別子として数値を用いる、請求項２０記載の情報処理装置。

【請求項２２】

前記ノード定義手段は前記ノード識別子として連続する整数を用いる、請求項２０記載の情報処理装置。

【請求項２３】

前記親子関係定義手段は、前記非ルート・ノードの各々に付与されたノード識別子と、前記関連付けられた親ノードに付与されたノード識別子と、の組の配列を前記記憶装置に格納する、請求項２０記載の情報処理装置。

【請求項２４】

ツリー型データ構造を記憶装置上に構築する情報処理装置であって、

同じ世代のノードよりも子ノードを優先して、ルート・ノードを含むノードに固有の連続する整数を付与するノード定義手段と、

前記ルート・ノード以外のノードである非ルート・ノードの各々に付与された整数の順に、前記非ルート・ノードの各々の親ノードに付与された整数を並べることにより形成される配列を前記記憶装置に格納する親子関係定義手段と、

を含む情報処理装置。

【請求項２５】

前記ノード定義手段は、

最初にルート・ノードに番号を付与する手段と、
既に番号が付与されたあるノードに唯一の子ノードが存在する場合には、当該子ノードに当該あるノードに付与された前記番号の次の番号を付与する手段と、
既に番号が付与されたあるノードに複数の子ノードが存在する場合には、当該複数の子ノードの間の兄弟関係に従って、弟ノードは直上の兄ノードの全ての子孫ノードに番号が付与された後に次の番号が付与されるように、一番上の兄ノードから一番下の弟ノードまで番号を付与する手段と、
を含む、
請求項２４記載の情報処理装置。
【請求項２６】
ツリー型データ構造を記憶装置上に構築する情報処理装置であって、
子ノードよりも同じ世代のノードを優先して、ルート・ノードを含むノードに固有の連続する整数を付与するノード定義手段と、
前記ルート・ノード以外のノードである非ルート・ノードの各々に付与された整数の順に、前記非ルート・ノードの各々の親ノードに付与された整数を並べることにより形成される配列を前記記憶装置に格納する親子関係定義手段と、
を含む情報処理装置。
【請求項２７】
前記ノード定義手段は、
各ノードが前記ルート・ノードから何世代目のノードであるか、及び、各世代に含まれるノード数を算出する手段と、
最初に前記ルート・ノードに番号を付与する手段と、
ある世代に含まれる全てのノードに番号が付与されたならば、当該ある世代の次の世代にノードが存在しなくなるまで、当該次の世代に含まれる全てのノードに対して、親ノードが異なる場合には、当該親ノードに番号が付与された順番に当該ノードに番号を付与し、当該親ノードが同一である場合には、当該親ノードから派生した複数の子ノードの間に兄弟関係を定義し、一番上の兄ノードから一番下の弟ノードまで直前に付与された番号の次の番号から連続的に変化する固有の整数を順に付与する手段と、
を含む、
請求項２６記載の情報処理装置。
【請求項２８】
前記配列から、あるノードに付与された整数以上の値が格納されている連続領域を抽出することにより、前記あるノードの全ての子孫ノードを特定する手段を更に有する、請求項２４又は２５記載の情報処理装置。
【請求項２９】
前記配列から、あるノードに付与された整数と同じ値が格納されている連続領域を抽出することにより、前記あるノードの全ての子ノードを特定する手段を更に有する、請求項２６又は２７記載の情報処理装置。
【請求項３０】
ツリー型データ構造を記憶装置上に構築する情報処理装置であって、
ルート・ノードから始めて全てのノードに連続的に変化する整数を一意に割り当てる手段と、
ノード間に親子関係を定義する手段と、
を含み、
前記全てのノードに整数を一意に割り当てる手段は、
同じ世代のノードよりも子ノードを優先して番号を付与する深さ優先モードと、子ノードよりも同じ世代のノードを優先して番号を付与する幅優先モードのどちらのモードでノードに番号を付与するかを選択する手段と、
前記深さ優先モードが選択された場合に、深さ優先でノードを検索し、検索された順にノードに番号を付与する手段と、

前記幅優先モードが選択された場合に、幅優先でノードを検索し、検索された順にノードに番号を付与する手段と、
を含み、
前記ノード間に親子関係を定義する手段は、子ノードに付与された番号の順に、当該子ノードに対応する親ノードに付与された番号を前記記憶装置に格納する手段を含む、
情報処理装置。
【請求項３１】
前記ノード間に親子関係を定義する手段は、
子ノードから親ノードへの関係を定義する子親表現モードと、親ノードから子ノードへの関係を定義する親子表現モードのどちらのモードで親子関係を定義するかを選択する手段と、
前記子親表現モードが選択された場合に、子ノードに付与された番号の順に、当該子ノードに対応する親ノードに付与された番号を前記記憶装置に格納する手段と、
前記親子表現モードが選択された場合に、親ノードに付与された番号の順に、当該親ノードに対応する子ノードに付与された番号の配列を前記記憶装置に格納する手段と、
を含む、
請求項３０記載の情報処理装置。
【請求項３２】
記憶装置上で親子関係を用いて表現されたツリー型データ構造の表現形式を変換する情報処理装置であって、
前記記憶装置は、子ノードに付与された番号の順に、当該子ノードに対応する親ノードに番号を付与することにより定義された前記親子関係を保持し、
同じ世代のノードよりも子ノードを優先してノードに番号を付与する深さ優先モードで表現されたツリー型データ構造の各ノードの世代を判定し、世代毎に属しているノードの個数を計数する手段と、
前記世代毎に属しているノードの個数に基づいて、子ノードよりも同じ世代のノードを優先してノードに番号を付与する幅優先モードで番号を付与する際に各世代において付与される番号を決定する手段と、
前記判定されたノードの世代及び前記決定された各世代において付与される番号に基づいて、前記各ノードの番号を前記幅優先モードで付与される番号に変換する変換配列を作成する手段と、
前記各ノードの前記親子関係を、前記変換配列を使用して前記幅優先モードで付与される番号で表現された親子関係に変換する手段と、
を有する情報処理装置。
【請求項３３】
記憶装置上で親子関係を用いて表現されたツリー型データ構造の表現形式を変換する情報処理装置であって、
前記記憶装置は、子ノードに付与された番号の順に、当該子ノードに対応する親ノードに番号を付与することにより定義された前記親子関係を保持し、
子ノードよりも同じ世代のノードを優先してノードに番号を付与する幅優先モードで表現されたツリー型データ構造の各ノードの子孫の数を計数する手段と、
親ノードに付与されるべき番号に、当該ノードと同一の親ノードから派生した兄弟ノードのうち当該ノードよりも前に番号が付与されている兄ノードの数及び当該兄ノードの子孫の数を加算することにより、前記幅優先モードで付与された番号を、同じ世代のノードよりも子ノードを優先してノードの番号を付与する深さ優先モードで付与される番号に変換する変換配列を作成する手段と、
前記各ノードの前記親子関係を、前記変換配列を使用して前記深さ優先モードで付与される番号で表現された親子関係に変換する手段と、
を有する情報処理装置。
【請求項３４】

記憶装置上で親子関係を用いて表現されたツリー型データ構造の表現形式を変換する情報処理装置であって、
前記記憶装置は、子ノードに付与された番号の順に、当該子ノードに対応する親ノードに番号を付与することにより定義された前記親子関係を保持し、
子ノードよりも同じ世代のノードを優先してノードに番号を付与する幅優先モードで表現されたツリー型データ構造の各ノードを深さ優先で検索し、前記幅優先モードで付与された番号を、同じ世代のノードよりも子ノードを優先してノードの番号を付与する深さ優先モードで付与される番号に変換する変換配列を作成する手段と、
前記各ノードの前記親子関係を、前記変換配列を使用して前記深さ優先モードで付与される番号で表現された親子関係に変換する手段と、
を有する情報処理装置。
【請求項３５】
記憶装置上で親子関係を用いて表現されたツリー型データ構造の表現形式を変換する情報処理装置であって、
前記記憶装置は、子ノードに付与された番号の順に、当該子ノードに対応する親ノードに付与された番号を、前記親子関係を定義する第１の配列の要素として保持し、
各ノードに関して、当該ノードに付与された番号が前記第１の配列の要素として出現する回数を計数する手段と、
前記各ノードに関して当該ノードに対応する子ノードに付与された番号を格納するため、前記計数された回数に応じた個数の連続領域を前記記憶領域に第２の配列として確保する手段と、
前記第１の配列の要素を順次に読み出し、当該要素と値が一致する番号が付与されたノードのために確保された前記第２の配列の要素として、前記第１の配列の要素に対する子ノードの番号を順次に格納する手段と、
を有する情報処理装置。
【請求項３６】
記憶装置上で親子関係を用いて表現されたツリー型データ構造の表現形式を変換する情報処理装置であって、
前記記憶装置は、親ノードに付与された番号の順に、当該親ノードに対応する子ノードに付与された番号を、前記親子関係を定義する第１の配列の要素として保持し、
子ノードに付与された番号の順に、当該子ノードに対応する親ノードに付与された番号を格納するため、前記記憶装置に第２の配列を確保する手段と、
前記第１の配列の要素を順次に読み出し、当該要素と値が一致する番号が付与されたノードのために確保された前記第２の配列の要素として、前記第１の配列の要素に対する親ノードの番号を順次に格納する手段と、
を有する情報処理装置。
【請求項３７】
ツリー型データ構造を記憶装置上に構築するコンピュータに、
ルート・ノードを含むノードに固有のノード識別子を付与するノード定義機能と、
前記ルート・ノード以外のノードである非ルート・ノードの各々に付与されたノード識別子に、前記非ルート・ノードの各々の親ノードに付与されたノード識別子を関連付ける親子関係定義機能と、
を実現させるためのプログラム。
【請求項３８】
前記ノード定義機能は前記ノード識別子として数値を用いる、請求項３７記載のプログラム。
【請求項３９】
前記ノード定義機能は前記ノード識別子として連続する整数を用いる、請求項３７記載のプログラム。
【請求項４０】

前記親子関係定義機能は、前記非ルート・ノードの各々に付与されたノード識別子と、前記関連付けられた親ノードに付与されたノード識別子と、の組の配列を前記記憶装置に格納する、請求項３７記載のプログラム。

【請求項４１】

ツリー型データ構造を記憶装置上に構築するコンピュータに、

同じ世代のノードよりも子ノードを優先して、ルート・ノードを含むノードに固有の連続する整数を付与するノード定義機能と、

前記ルート・ノード以外のノードである非ルート・ノードの各々に付与された整数の順に、前記非ルート・ノードの各々の親ノードに付与された整数を並べることにより形成される配列を前記記憶装置に格納する親子関係定義機能と、

を実現させるためのプログラム。

【請求項４２】

前記ノード定義機能は、

最初にルート・ノードに番号を付与する機能と、

既に番号が付与されたあるノードに唯一の子ノードが存在する場合には、当該子ノードに当該あるノードに付与された前記番号の次の番号を付与する機能と、

既に番号が付与されたあるノードに複数の子ノードが存在する場合には、当該複数の子ノードの間の兄弟関係に従って、弟ノードは直上の兄ノードの全ての子孫ノードに番号が付与された後に次の番号が付与されるように、一番上の兄ノードから一番下の弟ノードまで番号を付与する機能と、

を含む、請求項４１記載のプログラム。

【請求項４３】

ツリー型データ構造を記憶装置上に構築するコンピュータに、

子ノードよりも同じ世代のノードを優先して、ルート・ノードを含むノードに固有の連続する整数を付与するノード定義機能と、

前記ルート・ノード以外のノードである非ルート・ノードの各々に付与された整数の順に、前記非ルート・ノードの各々の親ノードに付与された整数を並べることにより形成される配列を前記記憶装置に格納する親子関係定義機能と、

を実現させるためのプログラム。

【請求項４４】

前記ノード定義機能は、

各ノードが前記ルート・ノードから何世代目のノードであるか、及び、各世代に含まれるノード数を算出する機能と、

最初に前記ルート・ノードに番号を付与する機能と、

ある世代に含まれる全てのノードに番号が付与されたならば、当該ある世代の次の世代にノードが存在しなくなるまで、当該次の世代に含まれる全てのノードに対して、親ノードが異なる場合には、当該親ノードに番号が付与された順番に当該ノードに番号を付与し、当該親ノードが同一である場合には、当該親ノードから派生した複数の子ノードの間に兄弟関係を定義し、一番上の兄ノードから一番下の弟ノードまで直前に付与された番号の次の番号から連続的に変化する固有の整数を順に付与する機能と、

を含む、請求項４３記載のプログラム。

【請求項４５】

前記配列から、あるノードに付与された整数以上の値が格納されている連続領域を抽出することにより、前記あるノードの全ての子孫ノードを特定する機能を更に有する、請求項４１又は４２記載のプログラム。

【請求項４６】

前記配列から、あるノードに付与された整数と同じ値が格納されている連続領域を抽出することにより、前記あるノードの全ての子ノードを特定する機能を更に有する、請求項４３又は４４記載のプログラム。

【請求項４７】

ツリー型データ構造を記憶装置上に構築するコンピュータに、
ルート・ノードから始めて全てのノードに連続的に変化する整数を一意に割り当てる機能と、
ノード間に親子関係を定義する機能と、
を実現させ、
前記全てのノードに整数を一意に割り当てる機能は、
同じ世代のノードよりも子ノードを優先して番号を付与する深さ優先モードと、子ノードよりも同じ世代のノードを優先して番号を付与する幅優先モードのどちらのモードでノードに番号を付与するかを選択する機能と、
前記深さ優先モードが選択された場合に、深さ優先でノードを検索し、検索された順にノードに番号を付与する機能と、
前記幅優先モードが選択された場合に、幅優先でノードを検索し、検索された順にノードに番号を付与する機能と、
を含み、
前記ノード間に親子関係を定義する機能は、子ノードに付与された番号の順に、当該子ノードに対応する親ノードに付与された番号を前記記憶装置に格納する機能を含む、
プログラム。
【請求項４８】
前記ノード間に親子関係を定義する機能は、
子ノードから親ノードへの関係を定義する子親表現モードと、親ノードから子ノードへの関係を定義する親子表現モードのどちらのモードで親子関係を定義するかを選択する機能と、
前記子親表現モードが選択された場合に、子ノードに付与された番号の順に、当該子ノードに対応する親ノードに付与された番号を前記記憶装置に格納する機能と、
前記親子表現モードが選択された場合に、親ノードに付与された番号の順に、当該親ノードに対応する子ノードに付与された番号の配列を前記記憶装置に格納する機能と、
を含む、請求項４７記載のプログラム。
【請求項４９】
記憶装置上で親子関係を用いて表現されたツリー型データ構造の表現形式を変換するコンピュータに、
子ノードに付与された番号の順に、当該子ノードに対応する親ノードに付与された番号を前記記憶装置に格納することにより前記親子関係を定義する機能と、
同じ世代のノードよりも子ノードを優先してノードに番号を付与する深さ優先モードで表現されたツリー型データ構造の各ノードの世代を判定し、世代毎に属しているノードの個数を計数する機能と、
前記世代毎に属しているノードの個数に基づいて、子ノードよりも同じ世代のノードを優先してノードに番号を付与する幅優先モードで番号を付与する際に各世代において付与される番号を決定する機能と、
前記判定されたノードの世代及び前記決定された各世代において付与される番号に基づいて、前記各ノードの番号を前記幅優先モードで付与される番号に変換する変換配列を作成する機能と、
前記各ノードの前記親子関係を、前記変換配列を使用して前記幅優先モードで付与される番号で表現された親子関係に変換する機能と、
を実現させるためのプログラム。
【請求項５０】
記憶装置上で親子関係を用いて表現されたツリー型データ構造の表現形式を変換するコンピュータに、
子ノードに付与された番号の順に、当該子ノードに対応する親ノードに付与された番号を前記記憶装置に格納することにより前記親子関係を定義する機能と、
子ノードよりも同じ世代のノードを優先してノードに番号を付与する幅優先モードで表

現されたツリー型データ構造の各ノードの子孫の数を計数する機能と、
親ノードに付与されるべき番号に、当該ノードと同一の親ノードから派生した兄弟ノードのうち当該ノードよりも前に番号が付与されている兄ノードの数及び当該兄ノードの子孫の数を加算することにより、前記幅優先モードで付与された番号を、同じ世代のノードよりも子ノードを優先してノードの番号を付与する深さ優先モードで付与される番号に変換する変換配列を作成する機能と、
前記各ノードの前記親子関係を、前記変換配列を使用して前記深さ優先モードで付与される番号で表現された親子関係に変換する機能と、
を実現させるためのプログラム。
【請求項５１】
記憶装置上で親子関係を用いて表現されたツリー型データ構造の表現形式を変換するコンピュータに、
子ノードに付与された番号の順に、当該子ノードに対応する親ノードに付与された番号を前記記憶装置に格納することにより前記親子関係を定義する機能と、
子ノードよりも同じ世代のノードを優先してノードに番号を付与する幅優先モードで表現されたツリー型データ構造の各ノードを深さ優先で検索し、前記幅優先モードで付与された番号を、同じ世代のノードよりも子ノードを優先してノードの番号を付与する深さ優先モードで付与される番号に変換する変換配列を作成する機能と、
前記各ノードの前記親子関係を、前記変換配列を使用して前記深さ優先モードで付与される番号で表現された親子関係に変換する機能と、
を実現させるためのプログラム。
【請求項５２】
記憶装置上で親子関係を用いて表現されたツリー型データ構造の表現形式を変換するコンピュータに、
子ノードに付与された番号の順に、当該子ノードに対応する親ノードに付与された番号を第１の配列の要素として前記記憶装置に格納することにより前記親子関係を定義する機能と、
各ノードに関して、当該ノードに付与された番号が前記第１の配列の要素として出現する回数を計数する機能と、
前記各ノードに関して当該ノードに対応する子ノードに付与された番号を格納するため、前記計数された回数に応じた個数の連続領域を前記記憶領域に第２の配列として確保する機能と、
前記第１の配列の要素を順次に読み出し、当該要素と値が一致する番号が付与されたノードのために確保された前記第２の配列の要素として、前記第１の配列の要素に対する子ノードの番号を順次に格納する機能と、
を実現させるためのプログラム。
【請求項５３】
記憶装置上で親子関係を用いて表現されたツリー型データ構造の表現形式を変換するコンピュータに、
親ノードに付与された番号の順に、当該親ノードに対応する子ノードに付与された番号を第１の配列の要素として前記記憶装置に格納することにより前記親子関係を定義する機能と、
子ノードに付与された番号の順に、当該子ノードに対応する親ノードに付与された番号を格納するため、前記記憶装置に第２の配列を確保する機能と、
前記第１の配列の要素を順次に読み出し、当該要素と値が一致する番号が付与されたノードのために確保された前記第２の配列の要素として、前記第１の配列の要素に対する親ノードの番号を順次に格納する機能と、
を実現させるためのプログラム。
【請求項５４】
請求項３７乃至５３のうちいずれか１項に記載のプログラムを記録したコンピュータ読

み取り可能な記録媒体。

【書類名】明細書
【発明の名称】ツリー型データ構造を取り扱う方法、情報処理装置、及び、プログラム
【技術分野】
【０００１】
本発明はツリー型データ構造を取り扱う方法、特に、ツリー型データ構造を表現し、記憶装置上に構築し、又は、変換する方法に関する。また、本発明は、かかる方法を実施する情報処理装置に関する。更に、本発明は、かかる方法を実行するためのプログラム、及び、このプログラムを記録した記録媒体に関する。
【背景技術】
【０００２】
データベースは種々の用途に用いられているが、中規模ないし大規模システムにおいては、論理的な矛盾が排除できるリレーショナルデータベース（ＲＤＢ）の使用が主流となっている。たとえば、ＲＤＢは飛行機の座席予約等のシステムに利用されている。この場合、キー項目を指定することにより、（多くの場合１件の）ターゲットを迅速に検索することもでき、或いは、予約の確定、キャンセル或いは変更などを行うことができる。また、各便の座席数はせいぜい数百であるため、特定の航空便の空席数を求めることも可能である。
【０００３】
このようなＲＤＢは、表形式データの取り扱いに適しているが、ツリー形式データの取り扱いには適していないことが知られている（例えば、非特許文献１を参照。）。
【０００４】
更に、アプリケーションの中には、表形式による表現よりもツリー形式による表現の方が適しているものが存在する。特に、近年、イントラネットやインターネットのアプリケーションのデータ標準として、ツリー型データ構造を採用するＸＭＬが普及している（ＸＭＬの詳細については、例えば、非特許文献２を参照。）。
【０００５】
しかし、ツリー型データ構造の取り扱い、例えば、ツリー形式データの検索は、一般に、大変効率が悪い。この効率の悪さの第１の理由は、データが各所のノードに分散して存在するため、データの存在すべき場所を直ちに特定することが困難である点にある。ＲＤＢでは、例えば、「年齢」というデータは、あるテーブルの「年齢」という項目だけに格納されている。しかし、ツリー型データ構造では、「年齢」というデータを保持するノードが各所に散在しているので、一般的には、ツリー型データ構造の全体を調べなければ、該当するデータを検索することができない。
【０００６】
効率の悪さの第２の理由は、検索の結果を表現するために時間がかかるという点にある。検索にヒットしたノード群を表現しようとすると、屡々、そのノードの子孫にあたるノードも表現しなければならないが、ＲＤＢＭＳとは異なりデータ構造が非定型であるため、子孫ノードを表現するために時間がかかる。
【０００７】
そこで、データベースの主流であるＲＤＢの利点をいかすため、従来、ツリー型データ構造をデータベース化するとき、ツリー形式データをＲＤＢ化する方法（例えば、特許文献１を参照。）が提案されている。ＲＤＢでは、データはテーブル（表）に分解して保持される。そのため、実際のツリー形式データをＲＤＢ化するには、ツリー形式データをテーブルに押し込める必要がある。しかし、様々のツリー型データ構造を取り扱うためには、その構造毎に個別にデータをテーブルに押し込め、システム設計を行わなければならない。したがって、ＲＤＢに基づくシステム構築は非常に手間のかかる作業である。
【０００８】
これに対して、ツリー形式データ、特に、ＸＭＬデータをそのままの形でデータベース化する方法も提案されている。ツリー型データ構造の場合、一つのノードに子孫ノードをぶら下げることができ、多様な表現が可能であるため、システム設計の手間を大幅に削減

することができる。したがって、ＸＭＬのようなツリー構造を取り扱える技術を核として、ツリー構造データを処理することへのニーズが高まっている。

【０００９】

ＸＭＬデータをそのままの形でデータベース化する方法の一例のアプローチは、ツリー構造に記入されているデータのコピーを取り出し、例えば、「年齢」という項目であれば、「年齢」の検索用インデックスデータを別途保持する（例えば、特許文献２を参照。）。これにより、データ自身に属性を付加できるというＸＭＬデータのメリットを十分に活用すると共に、タグを用いて表現された各項目の関係構造をそのまま記憶できるようにしている。

【特許文献１】特開２００３－２４８６１５号公報

【特許文献２】特開２００１－１９５４０６号公報

【非特許文献１】株式会社セック、"Karearea White Paper"、[online]、［平成１６年２月１９日検索］、インターネット＜URL:http://www.sec.co.jp/products/karearea/＞

【非特許文献２】W3C、"Extensible Markup Language (XML) 1.0 (Third Edition)"、[online]、２００４年２月４日、［平成１６年２月１９日検索］、インターネット＜URL:http://www.w3.org/TR/2004/REC-xml-20040204/＞

【発明の開示】

【発明が解決しようとする課題】

【００１０】

しかし、検索用インデックスデータを別途保持するようなアプローチでは、少なくともデータは二重に保持され、かつ、インデックスを作成するコスト及びインデックスを格納するためのデータ領域が必要となり、大規模なデータを保持する上で不利である。

【００１１】

実際、このようなメカニズムによって、実際に検索を行い、ノードを特定したとしても、そのノードを表現するためには時間がかかる。また、このメカニズムは、ノード間の関係を問題とする検索（例えば、祖先に「６０歳」という「年齢」を含み、子孫に「１歳」という「年齢」を含むツリーの抽出）には利用できない。

【００１２】

このような従来技術の根本的な問題点は、個々のデータのみに着目し、データを蓄えたノード間をポインタで接続することによりツリー型データ構造が表現されているため、データ間の関係、例えば、親子、祖先、子孫、兄弟（シブリング）、世代などの関係を効率的にトレースすることができないことにある。換言すると、ポインタは、その値が一定しないため、データの格納アドレスを示すという用途にしか使用できず、ノード間の関係を直接的に表現することができない。

【００１３】

そこで、本発明は、ツリー型データ構造のデータ間の関係を効率的にトレースすることができるツリー型データ構造の表現、構築、及び、変換方法の提供を目的とする。

【００１４】

更に、本発明は、ツリー型データ構造のデータ間の関係を効率的にトレースすることができるツリー型データ構造を構築、及び、変換する情報処理装置の提供を目的とする。

【００１５】

更に、本発明は、ツリー型データ構造のデータ間の関係を効率的にトレースすることができるツリー型データ構造の表現、構築、及び、変換プログラムの提供を目的とする。

【００１６】

更に、本発明は、上記ツリー型データ構造の表現、構築、及び、変換プログラムを記録した記録媒体の提供を目的とする。

【課題を解決するための手段】

【００１７】

上記目的を達成するため、本発明の原理は、ツリー型データ構造を構成するノード間の

親子関係を、親ノードに子ノードを対応付ける「親→子」関係ではなく、子ノードに親ノードを対応付ける「子→親」関係によって表現することである。

【００１８】

「親→子」関係によって親子関係を表現する場合、一つの親ノードに複数の子ノードが対応する場合があるので、親ノードと子ノードの二つの要素を特定しなければ親子関係を定義できない。即ち、親ノードを特定しても、その親ノードと親子関係にある子ノードを特定することができない。

【００１９】

これに対して「子→親」関係によって親子関係を表現する場合、一つの子ノードには必ず唯一の親ノードが対応するので、子ノードを特定することによって、この子ノードに対応する唯一の親ノードを直ちに特定することができる。

【００２０】

そのため、本発明によれば、ツリー型データ構造を構成するノード間の親子関係を記憶装置上で表現する方法は、請求項１に記載されるように、ルート・ノード以外のノードである非ルート・ノードの各々に対して、該非ルート・ノードの親ノードを関連付けることにより前記ノード間の親子関係を表現する。これにより、「子→親」表現された子のノードから親のノードのリストを辿ることでツリーのトポロジーを表現することができる。

【００２１】

また、本発明によれば、ツリー型データ構造を記憶装置上に構築する方法は、請求項２に記載されるように、

ルート・ノードを含むノードに固有のノード識別子を付与するノード定義ステップと、

前記ルート・ノード以外のノードである非ルート・ノードの各々に付与されたノード識別子に、前記非ルート・ノードの各々の親ノードに付与されたノード識別子を関連付ける親子関係定義ステップと、

を含む。このように、最初に、文字列、浮動小数、整数などの任意の識別情報によってノードにノード識別子を付与し、次に、「子→親」表現に基づいて親子関係を定義することによって、子ノードのノード識別子から親ノードのノード識別子を引く（ルックアップする）ことでツリーのトポロジーを表現することができる。

【００２２】

請求項２に記載された方法において、請求項３に記載されるように、前記ノード定義ステップは前記ノード識別子として数値を用いる。ノード識別子を特に数値で表すことによって、親のノード識別子の格納アドレスを特定することが容易になるので、より簡単に子のノード識別子から親のノード識別子を引くことができるようになる。

【００２３】

また、請求項２に記載された方法において、請求項４に記載されるように、前記ノード定義ステップは前記ノード識別子として連続する整数を用いる。ノード識別子を連続する整数で表すことによって、ノード識別子から、そのノードに対応する親ノードのノード識別子が格納されているアドレスを簡単に取得することができるので、子ノードのノード識別子から親ノードのノード識別子を引く処理を高速化することができる。

【００２４】

好ましい実施例によれば、ノード識別子は、０または１からの整数連番で表現される。

【００２５】

請求項１に記載された方法において、請求項５に記載されているように、前記ノード間の親子関係は、前記非ルート・ノードの各々と、前記関連付けられた親ノードと、の組の配列によって表現される。この結果として、配列を利用することにより、子ノードから親ノードを引く処理が高速化される。

【００２６】

請求項２に記載された方法において、請求項６に記載されているように、前記親子関係定義ステップは、前記非ルート・ノードの各々に付与されたノード識別子と、前記関連付けられた親ノードに付与されたノード識別子と、の組の配列を前記記憶装置に格納する。

この結果として、親ノードのノード識別子が格納されているアドレスを簡単に取得することができるようになるので、子ノードから親ノードを引く処理が高速化される。

【００２７】

ツリー型データ構造のノードにノード識別子として順序付きの番号を付与してノード間の親子関係を表現する場合、番号の付与順序に規則を定めることによって、その後のツリー型データ構造の取り扱いが容易になるという利点がある。本発明によれば、この番号の付与順序の規則として、同じ世代のノードよりも子ノードを優先する深さ優先モードと、子ノードよりも同じ世代のノードを優先する幅優先モードが利用される。

【００２８】

このため、本発明によれば、ツリー型データ構造を記憶装置上に構築する方法は、請求項７に記載されているように、

同じ世代のノードよりも子ノードを優先して、ルート・ノードを含むノードに固有の連続する整数を付与するノード定義ステップと、

前記ルート・ノード以外のノードである非ルート・ノードの各々に付与された整数の順に、前記非ルート・ノードの各々の親ノードに付与された整数を並べることにより形成される配列を前記記憶装置に格納する親子関係定義ステップと、

を含む。これにより、ノードは深さ優先で連続整数が付与され、ノード間の親子関係は「子→親」関係の配列によって表現される。深さ優先で連続番号が付与されたノードの親子関係を「子→親」関係に基づいて配列表現すると、あるノードの子孫ノードが連続領域に出現するという優れた性質が得られる。

【００２９】

本発明の好ましい一実施例では、請求項７に記載された方法において、請求項８に記載されているように、

前記ノード定義ステップは、

最初にルート・ノードに番号を付与するステップと、

既に番号が付与されたあるノードに唯一の子ノードが存在する場合には、当該子ノードに当該あるノードに付与された前記番号の次の番号を付与するステップと、

既に番号が付与されたあるノードに複数の子ノードが存在する場合には、当該複数の子ノードの間の兄弟関係に従って、弟ノードは直上の兄ノードの全ての子孫ノードに番号が付与された後に次の番号が付与されるように、一番上の兄ノードから一番下の弟ノードまで番号を付与するステップと、

を含む。これにより、深さ優先モードで同一の親ノードから派生した複数の子ノードの間に兄弟関係が定義される。

【００３０】

更に、本発明によれば、ツリー型データ構造を記憶装置上に構築する方法は、請求項９に記載されているように、

子ノードよりも同じ世代のノードを優先して、ルート・ノードを含むノードに固有の連続する整数を付与するノード定義ステップと、

前記ルート・ノード以外のノードである非ルート・ノードの各々に付与された整数の順に、前記非ルート・ノードの各々の親ノードに付与された整数を並べることにより形成される配列を前記記憶装置に格納する親子関係定義ステップと、

を含む。これにより、ノードは幅優先モードで連続整数が付与され、ノード間の親子関係は「子→親」関係の配列によって表現される。幅優先モードで連続番号が付与されたノードの親子関係を「子→親」関係に基づいて配列表現すると、親ノードに付与された番号が前記配列中に順序付き（昇順又は降順）で出現するという優れた性質が得られる。

【００３１】

本発明の好ましい一実施例では、請求項９に記載された方法において、請求項１０に記載されているように、

前記ノード定義ステップは、

各ノードが前記ルート・ノードから何世代目のノードであるか、及び、各世代に含まれ

るノード数を算出するステップと、
最初に前記ルート・ノードに番号を付与するステップと、
ある世代に含まれる全てのノードに番号が付与されたならば、当該ある世代の次の世代にノードが存在しなくなるまで、当該次の世代に含まれる全てのノードに対して、親ノードが異なる場合には、当該親ノードに番号が付与された順番に当該ノードに番号を付与し、当該親ノードが同一である場合には、当該親ノードから派生した複数の子ノードの間に兄弟関係を定義し、一番上の兄ノードから一番下の弟ノードまで直前に付与された番号の次の番号から連続的に変化する固有の整数を順に付与するステップと、
を含む。これにより、幅優先モードで同一の親ノードから派生した複数の子ノードの間に兄弟関係が定義される。

【００３２】

本発明の一実施例では、前記深さ優先モードの優れた性質を利用することにより、請求項７又は８記載の方法は、請求項１１に記載されているように、前記配列から、あるノードに付与された整数以上の値が格納されている連続領域を抽出することにより、前記あるノードの全ての子孫ノードを特定するステップを更に有する。これにより、あるノードの子孫ノードを表すノード群が前記配列内の連続ブロックとして獲得できる。

【００３３】

また、本発明の他の一実施例では、前記幅優先モードの優れた性質を利用することにより、請求項９又は１０記載の方法は、請求項１２に記載されているように、前記配列から、あるノードに付与された整数と同じ値が格納されている連続領域を抽出することにより、前記あるノードの全ての子ノードを特定するステップを更に有する。これにより、あるノードの子ノードを、例えば、二分探索で検索することが可能であり、即ち、$O(\log(n))$のオーダーで検索することが可能になる。

【００３４】

上述のように、ノードに連続番号を付与するための深さ優先モード及び幅優先モードは、それぞれ、固有の優れた性質を備えている。そこで、本発明によれば、ツリー型データ構造を記憶装置上に構築する方法は、請求項１３に記載されているように、
ルート・ノードから始めて全てのノードに連続的に変化する整数を一意に割り当てるステップと、
ノード間に親子関係を定義するステップと、
を含み、
前記全てのノードに整数を一意に割り当てるステップは、
同じ世代のノードよりも子ノードを優先して番号を付与する深さ優先モードと、子ノードよりも同じ世代のノードを優先して番号を付与する幅優先モードのどちらのモードでノードに番号を付与するかを選択するステップと、
前記深さ優先モードが選択された場合に、深さ優先でノードを検索し、検索された順にノードに番号を付与するステップと、
前記幅優先モードが選択された場合に、幅優先でノードを検索し、検索された順にノードに番号を付与するステップと、
を含み、
前記ノード間に親子関係を定義するステップは、子ノードに付与された番号の順に、当該子ノードに対応する親ノードに付与された番号を前記記憶装置に格納するステップを含む。これにより、深さ優先モードによるノード番号付与と幅優先モードによるノード番号付与を一つのシステムに併存させることができるので、状況に応じて適切な表現形式を利用することが可能である。

【００３５】

また、本発明の一実施例によれば、請求項１３に記載される方法において、請求項１４に記載されているように、
前記ノード間に親子関係を定義するステップは、
子ノードから親ノードへの関係を定義する子親表現モードと、親ノードから子ノードへ

の関係を定義する親子表現モードのどちらのモードで親子関係を定義するかを選択するステップと、
前記子親表現モードが選択された場合に、子ノードに付与された番号の順に、当該子ノードに対応する親ノードに付与された番号を前記記憶装置に格納するステップと、
前記親子表現モードが選択された場合に、親ノードに付与された番号の順に、当該親ノードに対応する子ノードに付与された番号の配列を前記記憶装置に格納するステップと、
を含む。これにより、「子→親」関係で表現されているノード間の親子関係を、状況に応じて、「親→子」関係で表現することが可能になる。「親→子」関係に基づく表現は、例えば、外部との情報交換の際に有利である。

【００３６】

上述のように、ツリー型データ構造の表現形式として、親子関係を表現するための「子→親」表現及び「親→子」表現、並びに、ノードに番号を付与するため深さ優先モード及び幅優先モードを選択的に利用可能である。そこで、本発明は、異なる表現形式の相互変換の方法を提供する。

【００３７】

本発明によれば、記憶装置上で親子関係を用いて表現されたツリー型データ構造の表現形式を変換する方法は、請求項１５に記載されているように、
前記親子関係は、子ノードに付与された番号の順に、当該子ノードに対応する親ノードに付与された番号を前記記憶装置に格納することにより定義され、
同じ世代のノードよりも子ノードを優先してノードに番号を付与する深さ優先モードで表現されたツリー型データ構造の各ノードの世代を判定し、世代毎に属しているノードの個数を計数するステップと、
前記世代毎に属しているノードの個数に基づいて、子ノードよりも同じ世代のノードを優先してノードに番号を付与する幅優先モードで番号を付与する際に各世代において付与される番号を決定するステップと、
前記判定されたノードの世代及び前記決定された各世代において付与される番号に基づいて、前記各ノードの番号を前記幅優先モードで付与される番号に変換する変換配列を作成するステップと、
前記各ノードの前記親子関係を、前記変換配列を使用して前記幅優先モードで付与される番号で表現された親子関係に変換するステップと、
を有する。これにより、深さ優先モードに基づく「子→親」表現形式から幅優先モードに基づく「子→親」表現形式への変換が可能になる。

【００３８】

また、本発明によれば、記憶装置上で親子関係を用いて表現されたツリー型データ構造の表現形式を変換する方法は、請求項１６に記載されているように、
前記親子関係は、子ノードに付与された番号の順に、当該子ノードに対応する親ノードに付与された番号を前記記憶装置に格納することにより定義され、
子ノードよりも同じ世代のノードを優先してノードに番号を付与する幅優先モードで表現されたツリー型データ構造の各ノードの子孫の数を計数するステップと、
親ノードに付与されるべき番号に、当該ノードと同一の親ノードから派生した兄弟ノードのうち当該ノードよりも前に番号が付与されている兄ノードの数及び当該兄ノードの子孫の数を加算することにより、前記幅優先モードで付与された番号を、同じ世代のノードよりも子ノードを優先してノードの番号を付与する深さ優先モードで付与される番号に変換する変換配列を作成するステップと、
前記各ノードの前記親子関係を、前記変換配列を使用して前記深さ優先モードで付与される番号で表現された親子関係に変換するステップと、
を有する。これにより、幅優先モードに基づく「子→親」表現形式から深さ優先モードに基づく「子→親」表現形式への高速変換が可能になる。

【００３９】

また、本発明によれば、記憶装置上で親子関係を用いて表現されたツリー型データ構造

の表現形式を変換する方法は、請求項１７に記載されているように、
前記親子関係は、子ノードに付与された番号の順に、当該子ノードに対応する親ノードに付与された番号を前記記憶装置に格納することにより定義され、
子ノードよりも同じ世代のノードを優先してノードに番号を付与する幅優先モードで表現されたツリー型データ構造の各ノードを深さ優先で検索し、前記幅優先モードで付与された番号を、同じ世代のノードよりも子ノードを優先してノードの番号を付与する深さ優先モードで付与される番号に変換する変換配列を作成するステップと、
前記各ノードの前記親子関係を、前記変換配列を使用して前記深さ優先モードで付与される番号で表現された親子関係に変換するステップと、
を有する。これにより、幅優先モードに基づく「子→親」表現形式から深さ優先モードに基づく「子→親」表現形式への検索に基づく変換が可能になる。「深さ優先」検索は、例えば、スタックを使用して番号変換配列を作成することにより実現される。
【００４０】
また、本発明によれば、記憶装置上で親子関係を用いて表現されたツリー型データ構造の表現形式を変換する方法は、請求項１８に記載されているように、
前記親子関係は、子ノードに付与された番号の順に、当該子ノードに対応する親ノードに付与された番号を第１の配列の要素として前記記憶装置に格納することにより定義され、
各ノードに関して、当該ノードに付与された番号が前記第１の配列の要素として出現する回数を計数するステップと、
前記各ノードに関して当該ノードに対応する子ノードに付与された番号を格納するため、前記計数された回数に応じた個数の連続領域を前記記憶領域に第２の配列として確保するステップと、
前記第１の配列の要素を順次に読み出し、当該要素と値が一致する番号が付与されたノードのために確保された前記第２の配列の要素として、前記第１の配列の要素に対する子ノードの番号を順次に格納するステップと、
を有する。これにより、親子関係は、「子→親」表現形式から「親→子」表現形式に変換される。即ち、変換後の親子関係は、親ノードに付与された番号の順に、当該親ノードに対応する子ノードに付与された番号を前記第２の配列の要素として前記記憶装置に格納することにより定義される。
【００４１】
更に、本発明によれば、記憶装置上で親子関係を用いて表現されたツリー型データ構造の表現形式を変換する方法は、請求項１９に記載されているように、
前記親子関係は、親ノードに付与された番号の順に、当該親ノードに対応する子ノードに付与された番号を第１の配列の要素として前記記憶装置に格納することにより定義され、
子ノードに付与された番号の順に、当該子ノードに対応する親ノードに付与された番号を格納するため、前記記憶装置に第２の配列を確保するステップと、
前記第１の配列の要素を順次に読み出し、当該要素と値が一致する番号が付与されたノードのために確保された前記第２の配列の要素として、前記第１の配列の要素に対する親ノードの番号を順次に格納するステップと、
を有する方法。これにより、親子関係は、「親→子」表現形式から「子→親」表現形式に変換される。即ち、変換後の親子関係は、子ノードに付与された番号の順に、当該子ノードに対応する親ノードに付与された番号を前記第２の配列の要素として前記記憶装置に格納することにより定義される。
【００４２】
また、本発明によれば、請求項２０乃至２２、及び、２３乃至３６に記載されているように、請求項２乃至４、及び、６乃至１９に係る方法を実施する情報処理装置が提供される。
【００４３】

また、本発明によれば、請求項３７乃至３９、及び、４０乃至５３に記載されているように、請求項２乃至４、及び、６乃至１９に係る方法を実行するためのプログラムが提供される。

【００４４】

更に、本発明によれば、請求項３７乃至５３に記載されたプログラムを記録した記録媒体が提供される。

【発明の効果】

【００４５】

本発明によれば、ツリー型データ構造のノード間の親子関係は、「子→親」表現に基づいて記述されるので、一つのノードに対して一つの格納場所を設けることにより親子関係を定義することができる。したがって、ツリー型データ構造を操作する際にアクセスされるメモリの量が低減し、これにより、操作も高速化される。

【００４６】

更に、本発明の「子→親」表現によれば、幅優先モードでノードに番号を付与することにより、あるノードから派生した子ノードを容易に検索することができる。

【００４７】

また、本発明の「子→親」表現によれば、深さ優先モードでノードに番号を付与することにより、あるノードの子孫ノードのブロックを容易に特定することができる。

【発明を実施するための最良の形態】

【００４８】

以下、添付図面を参照して、本発明の実施の形態につき説明を加える。

【００４９】

［コンピュータシステム構成］

図１は、本発明の実施の形態にかかるツリー型データ構造を取り扱うコンピュータシステムのハードウェア構成を示すブロックダイヤグラムである。図１に示すように、このコンピュータシステム１０は、通常のものと同様の構成であり、プログラムを実行することによりシステム全体および個々の構成部分を制御するＣＰＵ１２、ワークデータなどを記憶するＲＡＭ(Random Access Memory)１４、プログラム等を記憶するＲＯＭ(Read Only Memory)１６、ハードディスク等の固定記憶媒体１８、ＣＤ－ＲＯＭ１９をアクセスするためのＣＤ－ＲＯＭドライバ２０、ＣＤ－ＲＯＭドライバ２０や外部ネットワーク（図示せず）と接続された外部端子との間に設けられたインタフェース（Ｉ／Ｆ）２２、キーボードやマウスからなる入力装置２４、ＣＲＴ表示装置２６を備えている。ＣＰＵ１２、ＲＡＭ１４、ＲＯＭ１６、外部記憶媒体１８、Ｉ／Ｆ２２、入力装置２４および表示装置２６は、バス２８を介して相互に接続されている。

【００５０】

本実施の形態にかかる、ツリー型データ構造を記憶装置上に構築するプログラム、及び、ツリー型データ構造を記憶装置上で変換するプログラムは、ＣＤ－ＲＯＭ１９に収容され、ＣＤ－ＲＯＭドライバ２０に読取られても良いし、ＲＯＭ１６に予め記憶されていても良い。また、いったんＣＤ－ＲＯＭ１９から読み出したものを、外部記憶媒体１８の所定の領域に記憶しておいても良い。或いは、上記プログラムは、ネットワーク（図示せず）、外部端子およびＩ／Ｆ２２を経て外部から供給されるものであっても良い。

【００５１】

また、本発明の実施の形態にかかる情報処理装置は、コンピュータシステム１０にツリー型データ構造を記憶装置上に構築するプログラム、及び、ツリー型データ構造を記憶装置上で変換するプログラムを実行させることにより実現される。

【００５２】

［ツリー型データ構造］

図２は、ツリー形式データの一例であるＰＯＳデータの説明図であり、同図の（Ａ）は、このツリー形式データのデータ構造（即ち、トポロジー）及びデータ値を視覚的に表現した一例であり、同図の（Ｂ）は、同じツリー形式データをＸＭＬ形式で表現した一例で

ある。同図に示されるようにツリー型データ構造は、ルート・ノード（本例では、ＰＯＳデータ）から始めて、各ノードで枝分かれしてリーフ・ノード（端点）に至るノードとアークの組み合わせによって表現される。各ノードの実体的な値、例えば、店名ノードの値＝”フランス店”の格納場所は、店名ノードに関連したポインタで指定される。

【００５３】

本発明は、ツリー型データ構造のトポロジーを対象とするため、以下の説明では、主として、ツリー型データ構造のトポロジーに関して説明する。

【００５４】

従来、このようなツリー型データ構造は、データを蓄えたノード間をポインタで接続することによって表現されている。しかし、ポインタ表現は、ポインタ値に必然性がないという欠点がある。即ち、ある場合には特定のノードＡがある番地（例えば、１００番地）に格納され、別の場合には同じノードＡが別の番地（例えば、２００番地）に格納されるので、ポインタ値が一定ではなく、ポインタ値は、本質的にノードの格納アドレスを表現するに過ぎない。そのため、例えば、ノードが深さ優先の規則に従ってポインタで接続されている場合、これらのノードを幅優先の規則に従ってポインタで再接続することは困難である。

【００５５】

これに対して、本発明者は、ツリー型データ構造のトポロジーがアークリストによって記述可能であることに着目した。アークリストとは、ノード間の親子関係を表すアークのリストである。図３は、アークリストを用いたツリー型データ構造の表現形式の一例の説明図である。同図の例では、０、１０、２０、３０、４０、５０、６０、７０、８０、９０、１００及び１１０のノード識別子（ＩＤ）が付与された１２個のノードからなるツリー型データ構造が示されている。同図の（Ａ）はツリー型データ構造の全体を示している。同図において、丸形、ハート形などの図形の中央に記載された数字は、ノードＩＤを表し、矢印と矢印の側に記載された＜０，１０＞などの数字の対は、アークを表している。尚、ノードＩＤは、文字列には限られず、数値、特に、整数でもよい。同図の（Ｂ）は、親ノード（Ｆｒｏｍ－ＩＤ）から子ノード（Ｔｏ－ＩＤ）へのアークリストを示し、（Ｃ）は、ノードＩＤとノードＴｙｐｅの対のリストからなるノードリストを示す。尚、ツリー型データ構造を表現するだけの目的のためにはノードリストが無くても構わない。原理的には、このようなアークリストを用いることによって、ノード間の関係をポインタによらずに直接的に記述することが可能である。

【００５６】

［「子→親」関係に基づく表現］

図３の例では、アークリストは、親ノードに子ノードを対応付ける「親→子」関係に基づいて記述されている。そのため、一つの親ノード、例えば、ルート・ノード０には、３個の子ノード１０、６０及び８０が存在するため、アークリストのＦｒｏｍ－ＩＤには、同じノードＩＤの０が３回出現している。つまり、親ノードを特定しても子ノードを特定することができないので、アークリストは、要素Ｆｒｏｍ－ＩＤの配列と要素Ｔｏ－ＩＤの配列により構成される。アークリストを使用する場合、あるノードは、Ｆｒｏｍ－ＩＤの配列と、Ｔｏ－ＩＤの配列の両方の配列に出現する。

【００５７】

これに対して、親子関係は、「子→親」関係によっても表現することが可能である。この場合、ノード間の親子関係は、ルート・ノード以外のノードである非ルート・ノードの各々と、関連付けられた親ノードと、の組の配列によって表現される。この「子→親」関係によって親子関係を表現する場合、「親→子」関係の場合には得られなかった重要な性質がある。即ち、一つの子ノードには必ず唯一の親ノードが対応するので、子ノードを特定することによって、この子ノードに対応する唯一の親ノードを直ちに特定することができる。つまり、アークリストは、実際には、要素Ｔｏ－ＩＤの配列だけを準備すればよい。この結果として、アークリストを格納するための記憶容量が削減される。この記憶容量の削減は、メモリへのアクセス回数が低減するという効果があるので、結果的に、処理の

高速化が実現できる。

【００５８】

図４は、本発明の一実施例による「子→親」関係に基づくツリー型データ構造の表現方法の説明図である。同図の（Ａ）はツリー全体の説明図であり、（Ｂ）は「子→親」関係に基づくアークリストである。同図の（Ｂ）のアークリストは、ルート・ノードに対する親ノードの格納領域を含んでいるので、ルート・ノードの親ノードとして、便宜的に”－”が設定されている。但し、ルート・ノードに対応する親ノードは存在しないので、同図の（Ｃ）に示されるように、「子→親」関係に基づくアークリストからルート・ノードに対する親ノードの格納領域を除いても構わない。このように本発明の一実施例では、ルート・ノード以外のノードである非ルート・ノードの各々に対して、非ルート・ノードの親ノードを関連付けることによりノード間の親子関係を表現する。そして、「子→親」表現された子のノードから親のノードのリストを辿ることでツリーのトポロジーを表現することができる。

【００５９】

このような「子→親」関係に基づくツリー型データ構造は、本発明の一実施例によれば、図５に示されるように、図１に示されたコンピュータシステム１０に、ルート・ノードを含むノードに固有のノード識別子を付与するノード定義ステップ５０１と、前記ルート・ノード以外のノードである非ルート・ノードの各々に付与されたノード識別子に、前記非ルート・ノードの各々の親ノードに付与されたノード識別子を関連付ける親子関係定義ステップ５０２と、を実行させることによってＲＡＭ１４上に構築される。このように、最初に、文字列、浮動小数、整数などの任意の識別情報によってノードにノード識別子を付与し、次に、「子→親」表現に基づいて親子関係を定義することによって、子ノードのノード識別子から親ノードのノード識別子を引く（ルックアップする）ことでツリーのトポロジーを表現することができる。

【００６０】

［ノード識別子］

好ましい一実施例によれば、ノード定義ステップはノード識別子として数値を使用し、より好ましくは、連続する整数を使用し、更に好ましくは、０又は１からの整数連番を使用する。これにより、ノード識別子から、そのノードに対応する親ノードのノード識別子が格納されているアドレスを簡単に取得することができるので、子ノードのノード識別子から親ノードのノード識別子を引く処理を高速化することができる。

【００６１】

ツリー型データ構造のノードにノード識別子として順序付きの番号を付与してノード間の親子関係を表現する場合、番号の付与順序に規則を定めることによって、その後のツリー型データ構造の取り扱いが容易になるという利点がある。本発明によれば、この番号の付与順序の規則として、同じ世代のノードよりも子ノードを優先する深さ優先モードと、子ノードよりも同じ世代のノードを優先する幅優先モードが利用される。

【００６２】

図６は、本発明の一実施例によりＩＤ形式のツリー構造型データを整数連番形式のツリー構造型データへ変換する処理の説明図である。同図の（Ａ）には、各ノードにＩＤ番号が付与されたツリー構造型データが示され、（Ｂ）には、変換規則が示され、（Ｃ）には、各ノードに整数連番が付与されたツリー構造型データが示されている。本例の変換規則は、深さ優先で連続番号を付与する規則であり、具体的には、複数の子ノードが存在する場合、長子（一番上の兄）ノードに最小番号を付与し、末子（一番下の弟）ノードに大きい番号を付与し、かつ、兄弟ノードよりも子ノードを優先して番号を付与する。本例では、昇順に番号付けをしているが、降順に番号付けをしてもよい。

【００６３】

また、図７は、本発明の他の一実施例によりＩＤ形式のツリー構造型データを整数連番形式のツリー構造型データへ変換する処理の説明図である。同図の（Ａ）には、各ノードにＩＤ番号が付与されたツリー構造型データが示され、（Ｂ）には、変換規則が示され、

（Ｃ）には、各ノードに整数連番が付与されたツリー構造型データが示されている。本例の変換規則は、幅優先で連続番号を付与する規則であり、具体的には、複数の子ノードが存在する場合、長子（一番上の兄）ノードに最小番号を付与し、末子（一番下の弟）ノードに大きい番号を付与し、かつ、子ノードよりも兄弟ノードを優先して番号を付与する。本例では、昇順に番号付けをしているが、降順に番号付けをしてもよい。

【００６４】

このようにノード識別子として番号を使用すると、ノード番号から直ちに、即ち、Ｏ（１）のオーダーで、そのノードに関する格納値が格納されているアドレスを引くことができる。また、親子関係を「子→親」表現することによって、子ノードから親ノードを直ちに、即ち、Ｏ（１）のオーダーで引くことができる。

【００６５】

［深さ優先モード］

本発明の一実施例によれば、図６に示されるような深さ優先に基づくツリー型データ構造は、図１に示されたコンピュータシステム１０に、

同じ世代のノードよりも子ノードを優先して、ルート・ノードを含むノードに固有の連続する整数を付与するノード定義ステップと、

ルート・ノード以外のノードである非ルート・ノードの各々に付与された整数の順に、非ルート・ノードの各々の親ノードに付与された整数を並べることにより形成される配列を前記記憶装置に格納する親子関係定義ステップと、

を実行させることによって、記憶装置上に構築される。これにより、ノードは深さ優先で連続整数が付与され、ノード間の親子関係は「子→親」関係の配列によって表現される。

【００６６】

図８は、本発明の一実施例による深さ優先に基づくノード定義処理のフローチャートである。このノード定義処理は、コンピュータシステム１０に

最初にルート・ノードに番号を付与するステップ８０１と、

既に番号が付与されたあるノードに唯一の子ノードが存在する場合には、当該子ノードに当該あるノードに付与された前記番号の次の番号を付与するステップ８０２と、

既に番号が付与されたあるノードに複数の子ノードが存在する場合には、当該複数の子ノードの間の兄弟関係に従って、弟ノードは直上の兄ノードの全ての子孫ノードに番号が付与された後に次の番号が付与されるように、一番上の兄ノードから一番下の弟ノードまで番号を付与するステップ８０３と、

を実行させる。これにより、深さ優先モードで同一の親ノードから派生した複数の子ノードの間に兄弟関係が定義される。

【００６７】

図９は、本発明の一実施例により図６に示された深さ優先のツリー型データ構造から作成された「子→親」表現に基づく親子関係の配列の説明図である。同図にサブツリー１又はサブツリー２として示されているように、深さ優先で連続番号が付与されたノードの親子関係を「子→親」関係に基づいて配列表現すると、あるノードの子孫ノードが連続領域に出現するという優れた性質が得られる。

【００６８】

本発明の一実施例では、深さ優先モードの優れた性質を利用することにより、前記配列から、あるノードに付与された整数以上の値が格納されている連続領域を抽出することにより、前記あるノードの全ての子孫ノードを特定する。これにより、あるノードの子孫ノードを表すノード群が前記配列内の連続ブロックとして獲得できる。例えば、連続ブロックのサイズをｍとすると、あるノードの全ての子孫ノードを特定するための処理速度は、Ｏ（ｍ）のオーダーになる。

【００６９】

既に説明したように、ノード間の親子関係は、「子→親」関係の配列の他に、「親→子」関係の配列によっても表現できる。図１０は、図６に示された深さ優先のツリー型データ構造から作成された「親→子」表現に基づく親子関係の配列の説明図である。一つの親

ノードに対して複数の子ノードが存在し得るので、親子関係の配列は、各ノードに対する子ノードの番号が格納されている領域を示すための配列Ａｇｇｒと、子ノードの番号が格納されている配列Ｐ→Ｃの二つの配列により構成される。例えば、配列Ａｇｇｒの先頭から２番目の要素Ａｇｇｒ［１］の値は”３”であり、これは、ノード［１］に対する子ノードの番号は、配列Ｐ→Ｃの要素Ｐ→Ｃ［３］以降に格納されていることを表している。これにより、ノード［０］、即ち、ルート・ノードに対する子ノードは、配列Ｐ→Ｃの先頭から３個の要素、Ｐ→Ｃ［０］の１、Ｐ→Ｃ［１］の６、及びＰ→Ｃ［２］の８であることがわかる。

【００７０】

この「親→子」表現に基づく親子関係の配列の求め方を説明する。

（１）ノードの番号が配列Ｐ→Ｃの最大の添字（＝１１）と一致する場合、このノードに属する子ノードは存在しない。したがって、処理は継続されない。

（２）同図に太字で表された親ノードの番号からＡｇｇｒ値を求める。このＡｇｇｒ値は、配列Ｐ→Ｃの開始点を表す。

（３）太字で表された親ノード番号＋１に対応するＡｇｇｒ値を求める。このＡｇｇｒ値－１が配列Ｐ→Ｃの終了点である。

【００７１】

例えば、ノード０の子ノードの開始点は、Ａｇｇｒ［０］、即ち、０であり、終了点は、Ａｇｇｒ［１］－１、即ち、３－１＝２である。したがって、ノード０の子ノードは、配列Ｐ→Ｃの０～２番目の要素、即ち、１、６及び８である。

【００７２】

或いは、「親→子」表現に基づく親子関係は、より単純に、親ノード番号の配列と、対応する子ノード番号の配列と、の二つの配列により表現することも可能である。しかし、この配列を利用して親子関係を見つけるためには、親ノードの番号を検索しなければならないので、即ち、ｌｏｇ（ｎ）のアクセス時間を要するので効率が悪い。

【００７３】

［幅優先モード］

本発明の一実施例によれば、図７に示されるような幅優先に基づくツリー型データ構造は、図１に示されたコンピュータシステム１０に、

子ノードよりも同じ世代のノードを優先して、ルート・ノードを含むノードに固有の連続する整数を付与するノード定義ステップと、

前記ルート・ノード以外のノードである非ルート・ノードの各々に付与された整数の順に、前記非ルート・ノードの各々の親ノードに付与された整数を並べることにより形成される配列を前記記憶装置に格納する親子関係定義ステップと、

を実行させることによって、記憶装置上に構築される。これにより、ノードは幅優先モードで連続整数が付与され、ノード間の親子関係は「子→親」関係の配列によって表現される。

【００７４】

図１１は、本発明の一実施例による幅優先に基づくノート定義処理のフローチャートである。このノード定義処理は、コンピュータシステム１０に、

各ノードが前記ルート・ノードから何世代目のノードであるか、及び、各世代に含まれるノード数を算出するステップ１１０１と、

最初に前記ルート・ノードに番号を付与するステップ１１０２と、

ある世代に含まれる全てのノードに番号が付与されたならば、当該ある世代の次の世代にノードが存在しなくなるまで、当該次の世代に含まれる全てのノードに対して、親ノードが異なる場合には、当該親ノードに番号が付与された順番に当該ノードに番号を付与し、当該親ノードが同一である場合には、当該親ノードから派生した複数の子ノードの間に兄弟関係を定義し、一番上の兄ノードから一番下の弟ノードまで直前に付与された番号の次の番号から連続的に変化する固有の整数を順に付与するステップ１０１３と、

を実行させる。これにより、幅優先モードで同一の親ノードから派生した複数の子ノード

の間に兄弟関係が定義される。

【００７５】

図１２は、本発明の一実施例により図７に示された幅優先のツリー型データ構造から作成された「子→親」表現に基づく親子関係の配列の説明図である。同図に示されているように、幅優先で連続番号が付与されたノードの親子関係を「子→親」関係に基づいて配列表現すると、あるノードの子孫ノードは連続領域に出現するという優れた性質が得られる。これは、幅優先モードで連続番号が付与されたノードの親子関係を「子→親」関係に基づいて配列表現すると、親ノードに付与された番号が前記配列中に順序付き（昇順又は降順）で出現することによる。

【００７６】

したがって、本発明の一実施例では、幅優先モードの優れた性質を利用することにより、前記配列から、あるノードに付与された整数と同じ値が格納されている連続領域を抽出することにより、前記あるノードの全ての子ノードを特定する。これにより、あるノードの子ノードを、例えば、二分探索などの手法を用いて検索することが可能であり、即ち、Ｏ（ｌｏｇ（ｎ））のオーダーで検索することが可能になる。

【００７７】

既に説明したように、ノード間の親子関係は、「子→親」関係の配列の他に、「親→子」関係の配列によっても表現できる。図１３は、図７に示された幅優先のツリー型データ構造から作成された「親→子」表現に基づく親子関係の配列の説明図である。図１３一つの親ノードに対して複数の子ノードが存在し得るので、親子関係の配列は、各ノードに対する子ノードの番号が格納されている領域を示すための配列Ａｇｇｒと、子ノードの番号が格納されている配列Ｐ→Ｃの二つの配列により構成される。例えば、配列Ａｇｇｒの先頭から２番目の要素Ａｇｇｒ［１］の値は”３”であり、これは、ノード［１］に対する子ノードの番号は、配列Ｐ→Ｃの要素Ｐ→Ｃ［３］以降に格納されていることを表している。これにより、ノード［０］、即ち、ルート・ノードに対する子ノードは、配列Ｐ→Ｃの先頭から３個の要素、Ｐ→Ｃ［０］の１、Ｐ→Ｃ［１］の２、及び、Ｐ→Ｃ［２］の３であることがわかる。

【００７８】

この「親→子」表現に基づく親子関係の配列の求め方を説明する。

（１）ノードの番号が配列Ｐ→Ｃの最大の添字（＝１１）と一致する場合、このノードに属する子ノードは存在しない。したがって、処理は継続されない。

（２）同図に太字で表された親ノードの番号からＡｇｇｒ値を求める。このＡｇｇｒ値は、配列Ｐ→Ｃの開始点を表す。

（３）太字で表された親ノード番号＋１に対応するＡｇｇｒ値を求める。このＡｇｇｒ値－１が配列Ｐ→Ｃの終了点である。

【００７９】

例えば、ノード０の子ノードの開始点は、Ａｇｇｒ［０］、即ち、０であり、終了点は、Ａｇｇｒ［１］－１、即ち、３－１＝２である。したがって、ノード０の子ノードは、配列Ｐ→Ｃの０～２番目の要素、即ち、１、２及び３である。

【００８０】

［ツリー型データ構造の表現形式の相互変換］

上述のように、ノードに連続番号を付与するための深さ優先モード及び幅優先モードは、それぞれ、固有の優れた性質を備えている。そこで、本発明の一実施例によるコンピュータシステムは、深さ優先に基づく「子→親」表現形式と、幅優先に基づく「子→親」表現形式と、「親→子」表現形式と、の間で相互に表現形式を変換する。図１４は、本発明の一実施例による三つの表現形式の相互変換の関係を示す図である。

【００８１】

図１５は、本発明の一実施例によりコンピュータシステムによって実現されるツリー型データ構造の構築方法のフローチャートである。同図に示されるように、コンピュータシステム１０は、ルート・ノードから始めて全てのノードに連続的に変化する整数を一意に

割り当てるステップ１５１０と、ノード間に親子関係を定義するステップ１５２０と、を実行することにより、ツリー型データ構造を記憶装置上に構築する。

【００８２】

好ましくは、前記全てのノードに整数を一意に割り当てるステップ１５１０は、

同じ世代のノードよりも子ノードを優先して番号を付与する深さ優先モードと、子ノードよりも同じ世代のノードを優先して番号を付与する幅優先モードのどちらのモードでノードに番号を付与するかを選択するステップ１５１１と、

前記深さ優先モードが選択された場合に、深さ優先でノードを検索し、検索された順にノードに番号を付与するステップ１５１２と、

前記幅優先モードが選択された場合に、幅優先でノードを検索し、検索された順にノードに番号を付与するステップ１５１３と、

を含む。これにより、深さ優先モードによるノード番号付与と幅優先モードによるノード番号付与を一つのシステムに併存させることができるので、状況に応じて適切な表現形式を利用することが可能である。

【００８３】

また、好ましくは、前記ノード間に親子関係を定義するステップ１５２０は、

子ノードから親ノードへの関係を定義する子親表現モードと、親ノードから子ノードへの関係を定義する親子表現モードのどちらのモードで親子関係を定義するかを選択するステップ１５２１と、

前記子親表現モードが選択された場合に、子ノードに付与された番号の順に、当該子ノードに対応する親ノードに付与された番号を前記記憶装置に格納するステップ１５２２と、

前記親子表現モードが選択された場合に、親ノードに付与された番号の順に、当該親ノードに対応する子ノードに付与された番号の配列を前記記憶装置に格納するステップ１５２３と、

を含む。これにより、「子→親」関係で表現されているノード間の親子関係を、状況に応じて、「親→子」関係で表現することが可能になる。「親→子」関係に基づく表現は、例えば、外部との情報交換の際に有利である。

【００８４】

このように、本発明の一実施例によれば、ツリー型データ構造の表現形式として、親子関係を表現するための「子→親」表現及び「親→子」表現、並びに、ノードに番号を付与するため深さ優先モード及び幅優先モードを選択的に利用可能である。以下では、異なる表現形式の相互変換の方法を説明する。

【００８５】

［深さ優先「子→親」表現から幅優先「子→親」表現への変換］

図１６は、本発明の一実施例による（Ａ）深さ優先「子→親」表現から（Ｂ）幅優先「子→親」表現への変換の説明図である。図１７は、この本発明の一実施例による深さ優先「子→親」表現から幅優先「子→親」表現への変換方法のフローチャートである。親子関係は、子ノードに付与された番号の順に、当該子ノードに対応する親ノードに付与された番号をコンピュータシステム１０の記憶装置、例えば、ＲＡＭ１４に格納することにより定義されている。図１７に示されるように、コンピュータシステム１０は、

同じ世代のノードよりも子ノードを優先してノードに番号を付与する深さ優先モードで表現されたツリー型データ構造の各ノードの世代を判定し、世代毎に属しているノードの個数を計数するステップ１７０１と、

前記世代毎に属しているノードの個数に基づいて、子ノードよりも同じ世代のノードを優先してノードに番号を付与する幅優先モードで番号を付与する際に各世代において付与される番号を決定するステップ１７０２と、

前記判定されたノードの世代及び前記決定された各世代において付与される番号に基づいて、前記各ノードの番号を前記幅優先モードで付与される番号に変換する変換配列を作成するステップ１７０３と、

前記各ノードの前記親子関係を、前記変換配列を使用して前記幅優先モードで付与される番号で表現された親子関係に変換するステップ１７０４と、
を実行する。これにより、深さ優先モードに基づく「子→親」表現形式から幅優先モードに基づく「子→親」表現形式への変換が可能になる。

【００８６】

次に、上記ステップ１７０１～１７０４をより詳細に説明する。

【００８７】

ステップ１７０１では、世代毎のノード個数を計数する。図１８乃至２２は、本発明の一実施例による深さ優先に基づくツリー型データ構造の世代毎に属しているノードの個数を計数する処理の説明図である。最初に、図１８の（Ａ）手順０に示されるように、二つの変数配列を準備する。各ノードの世代を格納する配列ｄｅｐｔｈは配列Ｃ→Ｐと同じサイズである。世代毎のノード個数を格納する配列ｄｅｐｔｈ－ｃｏｕｎｔは、ツリー構造の深さの段数以上の適当なサイズであり、０で初期化されている。（Ｂ）手順１では、先頭の要素（具体的には、ルート・ノード）から始めて、ノードの世代（深さ）を判定して配列ｄｅｐｔｈに格納し、この要素の世代に属する要素の個数、即ち、配列ｄｅｐｔｈ－ｃｏｕｎｔの先頭の要素を１だけインクリメントする。ノード０は、世代が０であるため、ｄｅｐｔｈ［０］に０を設定し、ｄｅｐｔｈ－ｃｏｕｎｔ［０］を０から１にインクリメントする。図中、対象となるノードの番号は、太字で示されている。（Ｃ）手順２では、ノード１に対応する親ノードの番号を配列Ｃ→Ｐから取得し、親ノードの世代を調べる。配列Ｃ→Ｐ［１］の要素は０であるため、ｄｅｐｔｈ［０］を参照すると、要素は０であるので（図中、イタリック体で示されている）、親ノードの世代は０であることがわかる。ノード１の世代の値は、親ノードの世代の値に１を加えた値であるから、親ノードの世代の値＋１＝１である。よって、配列ｄｅｐｔｈ［１］に世代の値１を設定し、配列ｄｅｐｔｈ－ｃｏｕｎｔ［１］の要素を１だけインクリメントする。

【００８８】

以下、図１９の手順３～５、図２０の手順６～８、図２１の手順９～１１、及び、図２２の手順１２の順番に、ノード２～ノード１１に対し同様の処理を続ける。

【００８９】

手順３：次のＣ→Ｐの要素は１なので、ｄｅｐｔｈ［１］（親のｄｅｐｔｈに該当する）を参照する。親のｄｅｐｔｈは１であるので、自ノードのｄｅｐｔｈは、１＋１＝＞２になる。そこで自ノードのｄｅｐｔｈ（＝２）をｄｅｐｔｈ［２］に格納する。最後に、ｄｅｐｔｈ－ｃｏｕｎｔ［自ノードのｄｅｐｔｈ］をインクリメントする。

【００９０】

手順４：次のＣ→Ｐの要素は２なので、ｄｅｐｔｈ［２］（親のｄｅｐｔｈに該当する）を参照する。親のｄｅｐｔｈは２であるので、自ノードのｄｅｐｔｈは、２＋１＝＞３になる。そこで自ノードのｄｅｐｔｈ（＝３）をｄｅｐｔｈ［３］に格納する。最後に、ｄｅｐｔｈ－ｃｏｕｎｔ［自ノードのｄｅｐｔｈ］をインクリメントする。

【００９１】

手順５：次のＣ→Ｐの要素は２なので、ｄｅｐｔｈ［２］（親のｄｅｐｔｈに該当する）を参照する。親のｄｅｐｔｈは２であるので、自ノードのｄｅｐｔｈは、２＋１＝＞３になる。そこで自ノードのｄｅｐｔｈ（＝３）をｄｅｐｔｈ［４］に格納する。最後に、ｄｅｐｔｈ－ｃｏｕｎｔ［自ノードのｄｅｐｔｈ］をインクリメントする。

【００９２】

手順６：次のＣ→Ｐの要素は１なので、ｄｅｐｔｈ［１］（親のｄｅｐｔｈに該当する）を参照する。親のｄｅｐｔｈは１であるので、自ノードのｄｅｐｔｈは、１＋１＝＞２になる。そこで自ノードのｄｅｐｔｈ（＝２）をｄｅｐｔｈ［５］に格納する。最後に、ｄｅｐｔｈ－ｃｏｕｎｔ［自ノードのｄｅｐｔｈ］をインクリメントする。

【００９３】

手順７：次のＣ→Ｐの要素は０なので、ｄｅｐｔｈ［０］（親のｄｅｐｔｈに該当する）を参照する。親のｄｅｐｔｈは０であるので、自ノードのｄｅｐｔｈは、０＋１＝＞１

になる。そこで自ノードのｄｅｐｔｈ（＝１）をｄｅｐｔｈ［６］に格納する。最後に、ｄｅｐｔｈ－ｃｏｕｎｔ［自ノードのｄｅｐｔｈ］をインクリメントする。

【００９４】

手順８：次のＣ→Ｐの要素は６なので、ｄｅｐｔｈ［６］（親のｄｅｐｔｈに該当する）を参照する。親のｄｅｐｔｈは１であるので、自ノードのｄｅｐｔｈは、１＋１＝＞２になる。そこで自ノードのｄｅｐｔｈ（＝２）をｄｅｐｔｈ［７］に格納する。最後に、ｄｅｐｔｈ－ｃｏｕｎｔ［自ノードのｄｅｐｔｈ］をインクリメントする。

【００９５】

手順９：次のＣ→Ｐの要素は０なので、ｄｅｐｔｈ［０］（親のｄｅｐｔｈに該当する）を参照する。親のｄｅｐｔｈは０であるので、自ノードのｄｅｐｔｈは、０＋１＝＞１になる。そこで自ノードのｄｅｐｔｈ（＝１）をｄｅｐｔｈ［８］に格納する。最後に、ｄｅｐｔｈ－ｃｏｕｎｔ［自ノードのｄｅｐｔｈ］をインクリメントする。

【００９６】

手順１０：次のＣ→Ｐの要素は８なので、ｄｅｐｔｈ［８］（親のｄｅｐｔｈに該当する）を参照する。親のｄｅｐｔｈは０であるので、自ノードのｄｅｐｔｈは、１＋１＝＞２になる。そこで自ノードのｄｅｐｔｈ（＝２）をｄｅｐｔｈ［９］に格納する。最後に、ｄｅｐｔｈ－ｃｏｕｎｔ［自ノードのｄｅｐｔｈ］をインクリメントする。

【００９７】

手順１１：次のＣ→Ｐの要素は９なので、ｄｅｐｔｈ［９］（親のｄｅｐｔｈに該当する）を参照する。親のｄｅｐｔｈは２であるので、自ノードのｄｅｐｔｈは、２＋１＝＞３になる。そこで自ノードのｄｅｐｔｈ（＝３）をｄｅｐｔｈ［１０］に格納する。最後に、ｄｅｐｔｈ－ｃｏｕｎｔ［自ノードのｄｅｐｔｈ］をインクリメントする。

【００９８】

手順１２：次のＣ→Ｐの要素は９なので、ｄｅｐｔｈ［９］（親のｄｅｐｔｈに該当する）を参照する。親のｄｅｐｔｈは２であるので、自ノードのｄｅｐｔｈは、２＋１＝＞３になる。そこで自ノードのｄｅｐｔｈ（＝３）をｄｅｐｔｈ［１１］に格納する。最後に、ｄｅｐｔｈ－ｃｏｕｎｔ［自ノードのｄｅｐｔｈ］をインクリメントする。

【００９９】

これにより、図２３に示されるような配列ｄｅｐｔｈ及び配列ｄｅｐｔｈ－ｃｏｕｎｔが得られる。

【０１００】

次に、ステップ１７０２において、世代毎のノードの個数が格納されている配列ｄｅｐｔｈ－ｃｏｕｎｔの要素を累計数にする（即ち、個数を累計する）。例えば、配列（１，３，４，４，０）の要素の値を累計数化すると、

１→１

３→１＋３＝４

４→４＋４＝８

４→８＋４＝１２

０→１２＋０＝１２

により、（１，４，８，１２，１２）のようになる。この累計数から明らかであるように、世代０までのノード数は１個であり、世代１までのノード数は４個であり、世代２までのノード数は８個であり、世代３までのノード数は１２個であり、世代４までのノード数は１２個である。ここから、世代順にノードを並べた場合、世代０の先頭ノードは全体で０番目であり、世代１の先頭ノードは全体で１番目のノードであり、世代２の先頭ノードは全体で４番目のノードであり、世代３の先頭ノードは全体で８番目のノードであり、世代４の先頭ノードは全体で１２番目のノードであることがわかる。このように、世代毎のノード個数の配列ｄｅｐｔｈ－ｃｏｕｎｔの要素を累計数化することにより、世代順にノードを並べた場合の各世代の先頭ノードが全体で何番目のノードであるかを示す配列ｄｅｐｔｈ－ａｇｇｒが得られる。尚、好ましい一実施例では、配列ｄｅｐｔｈ－ａｇｇｒは、（１，４，８，１２，１２）そのままではなく、先頭に０を詰め、要素を一つずつ後へ

ずらすことにより得られる配列（０，１，４，８，１２）によって与えられる。この配列ｄｅｐｔｈ－ａｇｇｒは、幅優先モードで番号を付与する際に各世代において付与される番号を表している。

【０１０１】

ステップ１７０３では変換配列を作成する。図２４乃至２８は、本発明の一実施例によるノードの番号を幅優先モードで付与される番号に変換する変換配列を作成する処理の説明図である。最初に、図２４の（Ａ）手順０に示されるように、配列Ｃ→Ｐと同じサイズの整数配列である番号変換定義配列「Ｎｏ．の変換定義」の領域を確保する。次に、（Ｂ）手順１において、ノード０の世代、即ち、ｄｅｐｔｈ［０］を取り出し、その値０が指定する配列ｄｅｐｔｈ－ａｇｇｒの要素ｄｅｐｔｈ－ａｇｇｒ［０］の要素０を取り出す。この値０は、上述のように、世代０において付与される番号を表している。ノード０に対応する配列「Ｎｏ．の変換定義」の要素にこの値０を設定することにより、深さ優先モードで番号が付与されたノード０は、幅優先モードで番号を付与した場合に、ノード０に変換されることがわかる。

【０１０２】

図２４の（Ｂ）手順２では、ノード１の番号変換定義を行う。そのため、ノード１の世代を表すｄｅｐｔｈ［１］を取り出し、その値１が指定するｄｅｐｔｈ－ａｇｇｒ［１］の値１を取り出す。この値１を配列「Ｎｏ．の変換定義」［１］に設定すると共に、ｄｅｐｔｈ－ａｇｇｒ［１］の値を１だけ増加させる。要素が取り出されたｄｅｐｔｈ－ａｇｇｒ［１］の要素を１だけインクリメントすることにより、次に、世代１のノードが選択された場合、そのノードの変換後の番号は、ノード１の変換後の番号１よりも１だけ大きい番号、即ち、２になる。

【０１０３】

以下、図２５の手順３～５、図２６の手順６～８、図２７の手順９～１１、及び、図２８の手順１２の順番に、ノード２～ノード１１に対し同様の処理を続ける。

【０１０４】

例えば、手順３では、ｄｅｐｔｈ［２］を取り出し、その値が指定するｄｅｐｔｈ－ａｇｇｒの要素を取り出す。取り出したｄｅｐｔｈ－ａｇｇｒの該当する要素は、「Ｎｏ．の変換定義」に格納すると共に１だけ増加させて、ｄｅｐｔｈ－ａｇｇｒに格納する。手順４～１２では、それぞれ、ｄｅｐｔｈ［３］～ｄｅｐｔｈ［１１］を取り出し、その値が指定するｄｅｐｔｈ－ａｇｇｒの要素を取り出す。取り出したｄｅｐｔｈ－ａｇｇｒの該当する要素は、「Ｎｏ．の変換定義」に格納すると共に１だけ増加させて、ｄｅｐｔｈ－ａｇｇｒに格納する。

【０１０５】

これにより、図２８に示されるような最終的な配列「Ｎｏ．の変換定義」が得られる。

【０１０６】

ステップ１７０４では、各ノードの親子関係を、変換配列を使用して幅優先モードで付与される番号で表現された親子関係に変換する。図２９は、本発明の一実施例による深さ優先モードに基づく「子→親」表現形式の親子関係を幅優先モードに基づく「子→親」表現形式の親子関係に変換する処理の説明図である。例えば、深さ優先モードに基づく「子→親」表現形式で、子ノードＣと親ノードＰが関連付けられている場合、上記の最終的な番号変換定義配列「Ｎｏ．の変換定義」によって、ノード番号Ｃがノード番号Ｃ’に変換され、ノード番号Ｐがノード番号Ｐ’に変換されるならば、幅優先モードに基づく「子→親」表現形式では、子ノードＣ’と親ノードＰ’が関連付けられることになる。変換前の全ての子ノードＣに対して、変換後の子ノードの番号Ｃ’と変換後の親ノードの番号Ｐ’が得られたならば、格納位置がＣ’で表され、格納値がＰ’で表される親子関係の配列Ｃ’→Ｐ’が完成する。

【０１０７】

図２９の例では、最初に、格納位置（０，１，２，３，４，５，６，７，８，９，１０，１１）は、「Ｎｏ．の変換定義」配列を用いて、格納位置（０，１，４，８，９，５，

２，６，３，７，１０，１１）に変換される。これにより、同図に示されるように、格納位置の変換後の配列Ｃ→Ｐは、
格納位置の変換後の配列Ｃ→Ｐ［０］には、深さ優先による配列Ｃ→Ｐ［０］の値－１がセットされ、
格納位置の変換後の配列Ｃ→Ｐ［１］には、深さ優先による配列Ｃ→Ｐ［１］の値０がセットされ、
格納位置の変換後の配列Ｃ→Ｐ［２］には、深さ優先による配列Ｃ→Ｐ［６］の値０がセットされ、
格納位置の変換後の配列Ｃ→Ｐ［３］には、深さ優先による配列Ｃ→Ｐ［８］の値０がセットされ、
格納位置の変換後の配列Ｃ→Ｐ［４］には、深さ優先による配列Ｃ→Ｐ［２］の値１がセットされ、
格納位置の変換後の配列Ｃ→Ｐ［５］には、深さ優先による配列Ｃ→Ｐ［５］の値１がセットされ、
格納位置の変換後の配列Ｃ→Ｐ［６］には、深さ優先による配列Ｃ→Ｐ［７］の値６がセットされ、
格納位置の変換後の配列Ｃ→Ｐ［７］には、深さ優先による配列Ｃ→Ｐ［９］の値８がセットされ、
格納位置の変換後の配列Ｃ→Ｐ［８］には、深さ優先による配列Ｃ→Ｐ［３］の値２がセットされ、
格納位置の変換後の配列Ｃ→Ｐ［９］には、深さ優先による配列Ｃ→Ｐ［４］の値２がセットされ、
格納位置の変換後の配列Ｃ→Ｐ［１０］には、深さ優先による配列Ｃ→Ｐ［１０］の値９がセットされ、
格納位置の変換後の配列Ｃ→Ｐ［１１］には、深さ優先による配列Ｃ→Ｐ［１１］の値９がセットされる。

【０１０８】

次に、格納位置の変換後の配列Ｃ→Ｐの各要素の値を「Ｎｏ．の変換定義」配列を用いて変換することにより、格納値の変換後のＣ→Ｐ配列が得られる。

【０１０９】

図２９の例では、最初に格納位置の変換、即ち、子ノード番号の変換を行い、その後、格納値の変換、即ち、親ノード番号の変換を行っているが、格納値の変換を先に行った後に格納位置の変換を行ってもよく、或いは、格納位置の変換と格納値の変換を同時に行ってもよい。尚、ルート・ノードの親ノードを表すノード番号”－１”は、変換する必要がない。

【０１１０】

以上の処理により、図１６に示されるような深さ優先「子→親」表現から幅優先「子→親」表現への変換が行われる。

【０１１１】

［幅優先「子→親」表現から深さ優先「子→親」表現への高速変換］

図３０は、本発明の一実施例による（Ａ）幅優先「子→親」表現から（Ｂ）深さ優先「子→親」表現への変換の説明図である。図３１は、この本発明の一実施例による幅優先「子→親」表現から深さ優先「子→親」表現への変換方法のフローチャートである。親子関係は、子ノードに付与された番号の順に、当該子ノードに対応する親ノードに付与された番号をコンピュータシステム１０の記憶装置、例えば、ＲＡＭ１４に格納することにより定義されている。図３１に示されるように、コンピュータシステム１０は、

子ノードよりも同じ世代のノードを優先してノードに番号を付与する幅優先モードで表現されたツリー型データ構造の各ノードの子孫の数を計数するステップ３１０１と、

親ノードに付与されるべき番号に、当該ノードと同一の親ノードから派生した兄弟ノードのうち当該ノードよりも前に番号が付与されている兄ノードの数及び当該兄ノードの子

孫の数を加算することにより、前記幅優先モードで付与された番号を、同じ世代のノードよりも子ノードを優先してノードの番号を付与する深さ優先モードで付与される番号に変換する変換配列を作成するステップ３１０２と、

前記各ノードの前記親子関係を、前記変換配列を使用して前記深さ優先モードで付与される番号で表現された親子関係に変換するステップ３１０３と、

を実行する。これにより、幅優先モードに基づく「子→親」表現形式から深さ優先モードに基づく「子→親」表現形式への高速変換が可能になる。

【０１１２】

次に、上記ステップ３１０１～３１０３をより詳細に説明する。

【０１１３】

ステップ３１０１では、ノードの子孫の数を計数する。図３２乃至３３は、本発明の一実施例による幅優先に基づくツリー型データ構造の各ノードの子孫の個数を計数する処理の説明図である。最初に、図３２に示されるように、配列Ｃ→Ｐと同じサイズのノード数配列を作成し、ノード数配列の各要素を１で初期化する。

【０１１４】

次に、図３３に示されるように、世代を遡る順番に（即ち、ルート・ノードへ向かって）、配列Ｃ→Ｐの要素の値を使ってノード数配列にアクセスし、配列Ｃ→Ｐの子ノードに対応するノード数配列の要素の値を、配列Ｃ→Ｐの親ノードに対応するノード数配列の要素の値に加算する。これにより、親ノードに対応するノード数配列の要素には、その親ノードの子孫ノードの個数が加算される。例えば、図３３の手順１では、配列Ｃ→Ｐ［１１］の値から、子ノード１１の親ノードはノード７であることがわかる。そこで、子ノード１１に対応するノード数配列の要素、即ち、ノード数配列［１１］の値を、親ノード７に対応するノード数配列の要素、即ち、ノード数配列［７］の値に加算する。これにより、子ノード１１の子孫ノードの数が親ノード７の子孫ノードの数に含まれる。手順２乃至１１に従って同様の処理を行うと、図３３に示されるような、ノード数の加算により得られたノード数配列が作成される。尚、本例では、配列Ｃ→Ｐの先頭の要素である配列Ｃ→Ｐ［０］は、ルート・ノードに対応しているので、ノード数の加算は、配列の要素Ｃ→Ｐ［１］に対する手順１１の処理で終了する。既に説明したように、ルート・ノードを配列Ｃ→Ｐから除いても構わない。最終的なノード数配列の要素の値は、自ノードもカウントされているので、例えば、ノード０の子孫ノードの個数は、（１２－１）＝１１個である。

【０１１５】

ステップ３１０２では、幅優先モードで番号が付与されたノード毎に、親ノードの番号に、兄ノードの数と兄ノードの子孫の数を加算することにより、当該ノードの幅優先モードによるノードの番号を深さ優先モードの番号に変換する変換配列を作成する。上述のように、幅優先モードは、子ノードよりも同じ世代のノードを優先してノードに番号を付与するものであり、深さ優先モードは、同じ世代のノードよりも子ノードを優先してノードの番号を付与するものである。そのため、ある親ノードに唯一の子ノードが存在する場合、深さ優先モードでは、この子ノードには、親ノードの次の番号が付与される。一方、ある親ノードに複数の子ノードが存在する場合、ある子ノードの番号は、その子ノードの兄ノード及び兄ノードの全ての子孫ノードに番号が付与された後に付与される。ある兄ノードの全ての子孫ノードの個数は、ノード数配列を参照することにより得ることができる。

【０１１６】

図３４は、本発明の一実施例による幅優先モードの番号から深さ優先モードの番号への変換配列を作成する処理の説明図である。この処理は、ルート・ノード０から順番に進められる。手順１では、配列Ｃ→Ｐにおいて、ルート・ノード０の参照先、即ち、親ノードは存在しないので、ルート・ノードの番号は０のままである。そこで、ルート・ノードに対するノード数配列の要素の値は０に書き換えられる。

【０１１７】

次に、手順１では、配列Ｃ→Ｐの要素の値が０であるノード、即ち、ノード０を親ノードとしてもつノードが処理の対象となる。本例では、ノード１、ノード２及びノード３は

、共通の親ノード０から派生した子ノードであるので、兄弟ノードと呼ばれる。幅優先モードでは、兄弟ノードの中に長子から末子へ向かって、ノード１、ノード２、ノード３の順序が付けられている。長子であるノード１は、兄ノードが存在しないので、親ノードであるノード０の次の番号を付与すればよい。したがって、ノード１に対するノード数配列の要素の値は、５から１に書き換えられる。次に、ノード２は、兄ノードであるノード１が存在するので、ノード１以下のノードの個数５を、親ノードのノード番号０に加算し、更に、自ノードの個数分の１を加算して得られるノード番号６に変換される。末子のノード３は、兄ノード１と兄ノード２が存在するので、兄ノード１以下のノードの個数５と、兄ノード２以下のノードの個数２と、親ノードのノード番号０と、自ノードの個数１と、を加算することにより得られるノード番号８に変換される。

【０１１８】

手順２から手順７まで、同一の親から派生した兄弟ノードの単位で手順１と同様の処理を繰り返すことにより、ノード数配列は変換配列に書き換えられる。勿論、ノード数配列とは別に変換配列を作成しても構わない。

【０１１９】

ステップ３１０３では、各ノードの親子関係を、変換配列を使用して深さ優先モードで付与される番号で表現された親子関係に変換する。図３５は、本発明の一実施例による幅優先モードに基づく「子→親」表現形式の親子関係を深さ優先モードに基づく「子→親」表現形式の親子関係に変換する処理の説明図である。この処理は、図２９に関して説明した処理と同様の処理である。例えば、幅優先モードに基づく「子→親」表現形式で、子ノードＣと親ノードＰが関連付けられている場合、上記の変換配列（「Ｎｏ．の変換定義」）によって、ノード番号Ｃがノード番号Ｃ’に変換され、ノード番号Ｐがノード番号Ｐ’に変換されるならば、深さ優先モードに基づく「子→親」表現形式では、子ノードＣ’と親ノードＰ’が関連付けられることになる。変換前の全ての子ノードＣに対して、変換後の子ノードの番号Ｃ’と変換後の親ノードの番号Ｐ’が得られたならば、格納位置がＣ’で表され、格納値がＰ’で表される親子関係の配列Ｃ’→Ｐ’が完成する。

【０１２０】

図３５の例では、最初に格納位置の変換、即ち、子ノード番号の変換を行い、その後、格納値の変換、即ち、親ノード番号の変換を行っているが、格納値の変換を先に行った後に格納位置の変換を行ってもよく、或いは、格納位置の変換と格納値の変換を同時に行ってもよい。尚、ルート・ノードの親ノードを表すノード番号”－１”は、変換する必要がない。

【０１２１】

以上の処理により、図３０に示されるような幅優先「子→親」表現から深さ優先「子→親」表現への高速変換が行われる。

【０１２２】

［幅優先「子→親」表現から深さ優先「子→親」表現への変換］

本発明の他の一実施例によれば、幅優先「子→親」表現から深さ優先「子→親」表現への変換は、図３０乃至３５に関して説明した高速変換方法の他に、検索を利用した変換方法によっても実現できる。以下では、図３０に示された例を用いて、この検索を利用した変換方法を説明する。

【０１２３】

幅優先「子→親」表現による親子関係は、子ノードに付与された番号の順に、当該子ノードに対応する親ノードに付与された番号をコンピュータシステム１０の記憶装置、例えば、ＲＡＭ１４に格納することにより定義されている。コンピュータシステム１０は、

子ノードよりも同じ世代のノードを優先してノードに番号を付与する幅優先モードで表現されたツリー型データ構造の各ノードを深さ優先で検索し、前記幅優先モードで付与された番号を、同じ世代のノードよりも子ノードを優先してノードの番号を付与する深さ優先モードで付与される番号に変換する変換配列を作成するステップと、

前記各ノードの前記親子関係を、前記変換配列を使用して前記深さ優先モードで付与さ

れる番号で表現された親子関係に変換するステップと、
を実行する。

【０１２４】

図３６乃至４３は、本発明の他の一実施例による幅優先「子→親」表現から深さ優先「子→親」表現への変換方法の説明図である。幅優先モードに基づく「子→親」表現形式から深さ優先モードに基づく「子→親」表現形式への検索に基づく変換において、「深さ優先」検索は、例えば、スタックを使用して番号変換配列を作成することにより実現される。

【０１２５】

変換配列を作成するステップの第１段階では、図３６に示されるような変数を作成する。変換配列は、配列Ｃ→Ｐと同じサイズの整数配列であり、－１で初期化される。変数ＣＵＲＲＥＮＴ＿ＮＯ及び変数ＳＴＡＣＫ＿ＰＯＳは０で初期化される。配列ＳＴＡＣＫは適当な長さの整数配列である。

【０１２６】

変換配列を作成するステップの第２段階では、図３７乃至４３の手順１～手順２０に示されるように、トレースを行いながら変換配列を作成する。

【０１２７】

手順１：（１）Ｃ→Ｐの要素で－１を含む場所（ルート・ノードを示す）を探し、添字０＝０の場所を見つける。（２）添字（＝０）をＳＴＡＣＫに格納し、同時に、ＳＴＡＣＫ＿ＰＯＳをインクリメントする。（３）添字（＝０）と同じ位置の変換配列の要素にＣＵＲＲＥＮＴ＿ＮＯの値を格納し、ＣＵＲＲＮＴ＿ＮＯをインクリメントする。

【０１２８】

手順２：（１）ＳＴＡＣＫ＿ＰＯＳ－１のスタック格納値（＝０）を指す、未使用で最小のＣ→Ｐの位置を検索する。未使用か否かは、変換配列の要素が１であるかどうかで判定できる。未使用かつ最小の添字は（＝１）に存在することが分かるので、その位置にポインタ（矢印マーク）を設定する。（２）この添字（＝１）をＳＴＡＣＫに格納し、ＳＴＡＣＫ＿ＰＯＳをインクリメントする。（３）添字（＝１）と同じ位置の変換配列の要素にＣＵＲＲＥＮＴ＿ＮＯの値を格納し、ＣＵＲＲＮＴ＿ＮＯをインクリメントする。

【０１２９】

手順３：（１）ＳＴＡＣＫ＿ＰＯＳ－１のスタック格納値（＝１）を指す、未使用で最小のＣ→Ｐの位置を検索する。未使用か否かは、変換配列の要素が１であるかどうかで判定できる。未使用かつ最小の添字は（＝４）に存在することが分かるので、その位置にポインタ（矢印マーク）を設定する。（２）この添字（＝４）をＳＴＡＣＫに格納し、ＳＴＡＣＫ＿ＰＯＳをインクリメントする。（３）添字（＝４）と同じ位置の変換配列の要素にＣＵＲＲＥＮＴ＿ＮＯの値を格納し、ＣＵＲＲＮＴ＿ＮＯをインクリメントする。

【０１３０】

手順４：（１）ＳＴＡＣＫ＿ＰＯＳ－１のスタック格納値（＝４）を指す、未使用で最小のＣ→Ｐの位置を検索する。未使用か否かは、変換配列の要素が１であるかどうかで判定できる。未使用かつ最小の添字は（＝８）に存在することが分かるので、その位置にポインタ（矢印マーク）を設定する。（２）この添字（＝８）をＳＴＡＣＫに格納し、ＳＴＡＣＫ＿ＰＯＳをインクリメントする。（３）添字（＝８）と同じ位置の変換配列の要素にＣＵＲＲＥＮＴ＿ＮＯの値を格納し、ＣＵＲＲＮＴ＿ＮＯをインクリメントする。

【０１３１】

手順５：（１）ＳＴＡＣＫ＿ＰＯＳ－１のスタック格納値（＝８）を指す、未使用で最小のＣ→Ｐの位置を検索する。未使用か否かは、変換配列の要素が１であるかどうかで判定できる。未使用かつ最小の添字は存在しないことがわかるので、ＳＴＡＣＫ＿ＰＯＳをデクリメントする。

【０１３２】

手順６：（１）ＳＴＡＣＫ＿ＰＯＳ－１のスタック格納値（＝４）を指す、未使用で最小のＣ→Ｐの位置を検索する。未使用か否かは、変換配列の要素が１であるかどうかで判

定できる。未使用かつ最小の添字は（＝９）に存在することが分かるので、その位置にポインタ（矢印マーク）を設定する。（２）この添字（＝９）をＳＴＡＣＫに格納し、ＳＴＡＣＫ＿ＰＯＳをインクリメントする。（３）添字（＝９）と同じ位置の変換配列の要素にＣＵＲＲＥＮＴ＿ＮＯの値を格納し、ＣＵＲＲＮＴ＿ＮＯをインクリメントする。

【０１３３】

以下同様に手順７～手順２０を実行する。手順２０の終了時点で、ＣＵＲＲＥＮＴ＿ＮＯが変換配列のサイズ（＝１２）に達し、変換配列が完成したことがわかる。ここで、変換配列作成処理は終了する。

【０１３４】

この検索に基づく変換方法によって作成された変換配列は、先の高速変換方法の例で作成された図３４の変換配列と同じであることがわかる。

【０１３５】

次に、図３５を参照して説明した本発明の一実施例による幅優先モードに基づく「子→親」表現形式の親子関係を深さ優先モードに基づく「子→親」表現形式の親子関係に変換する処理と全く同様の処理によって、親子関係の表現を変換する。

【０１３６】

［「子→親」表現から「親→子」表現への変換］

次に、本発明の一実施例による子ノードに親ノードを対応付ける「子→親」関係から親ノードに子ノードを対応付ける「親→子」関係への変換方法を説明する。

【０１３７】

図４４は、本発明の一実施例による「子→親」表現から「親→子」表現への変換方法のフローチャートである。親子関係は、子ノードに付与された番号の順に、当該子ノードに対応する親ノードに付与された番号を第１の配列の要素としてコンピュータシステム１０の記憶装置、例えば、ＲＡＭ１４に格納することにより定義されている。コンピュータシステム１０は、

各ノードに関して、当該ノードに付与された番号が前記第１の配列の要素として出現する回数を計数するステップ４４０１と、

前記各ノードに関して当該ノードに対応する子ノードに付与された番号を格納するため、前記計数された回数に応じた個数の連続領域を前記記憶領域に第２の配列として確保するステップ４４０２と、

前記第１の配列の要素を順次に読み出し、当該要素と値が一致する番号が付与されたノードのために確保された前記第２の配列の要素として、前記第１の配列の要素に対する子ノードの番号を順次に格納するステップ４４０３と、

を実行する。これにより、親子関係は、「子→親」表現形式から「親→子」表現形式に変換される。即ち、変換後の親子関係は、親ノードに付与された番号の順に、当該親ノードに対応する子ノードに付与された番号を前記第２の配列の要素として前記記憶装置に格納することにより定義される。

【０１３８】

この変換方法では、深さ優先又は幅優先の性質はそのまま保存されるので、深さ優先モードに基づく「子→親」表現は、深さ優先モードに基づく「親→子」表現に変換され、幅優先モードに基づく「子→親」表現は、幅優先モードに基づく「親→子」表現に変換される。以下では、深さ優先モードに基づく「子→親」表現から深さ優先モードに基づく「親→子」表現への変換の例を説明する。図４５は、深さ優先モードに基づくツリー型データ構造の一例の説明図であり、同図の（Ａ）はツリー型データ構造の全体を示し、（Ｂ）は「子→親」表現形式による親子関係を示し、（Ｃ）は「親→子」表現形式による親子関係を示している。本実施例では、（Ｂ）のような表現形式を（Ｃ）のような表現形式へ変換する。

【０１３９】

図４６乃至４７は、本発明の一実施例による深さ優先モードに基づく「子→親」表現から深さ優先モードに基づく「親→子」表現への変換方法の説明図である。

【０１４０】
図４６の（Ａ）手順１では、最初に、変換後の「親→子」表現形式の親子関係を格納するための領域を確保し、初期化する。既に説明したように、「親→子」表現形式の場合、配列Ａｇｇｒと配列Ｐ→Ｃを用意する。配列Ａｇｇｒは、各ノードに対する子ノードの番号が格納されている領域を示すための配列であり、配列Ｐ→Ｃは子ノードの番号を格納するための配列である。配列Ａｇｇｒのサイズは、（「子→親」表現形式の）配列Ｃ→Ｐと同じサイズであり、配列Ａｇｇｒは０で初期化する。配列Ｐ→Ｃのサイズは、配列Ａｇｇｒよりも要素１個分小さいサイズで足りる。配列Ｐ→Ｃは、特に初期化する必要はないが、図４６では、理解しやすいように－１で初期化されている。
【０１４１】
図４６の（Ｂ）手順２では、配列Ｃ→Ｐの各要素が指定する配列Ａｇｇｒの要素を１ずつインクリメントする。配列Ｃ→Ｐの要素の値は、親ノードの番号を表しているので、カウントアップされた配列Ａｇｇｒの各要素は、配列Ａｇｇｒの添字の番号と一致するノードの子ノードの個数を表す。
【０１４２】
図４６の（Ｃ）手順３では、カウントアップされたＡｇｇｒの要素の値を累計数に変換する。これにより、各ノードに対する子ノードの番号が格納されている領域を示すための配列Ａｇｇｒが完成する。尚、本例では、累計数にするときに、１つずつ後へシフトさせている。
【０１４３】
図４７の（Ａ）手順４では、配列Ｃ→Ｐから配列Ｐ→Ｃへノード番号を転送する。配列Ｃ→Ｐの先頭の要素は、ルート・ノード０についての情報であり、ルート・ノード０には親ノードが定義されていないので（本例では、配列Ｃ→Ｐの要素が負の値－１であるので何もしない）、実際の処理は、ノード１に対応する配列Ｃ→Ｐの添字１から始まる。配列Ｃ→Ｐの添字１の要素は、ノード１の親ノードがノード０であることを示している。そこで、配列Ａｇｇｒの中でノード０に対する要素、即ち、添字１に対応する要素を参照すると、値０が格納されているので、配列Ｐ→Ｃの中でこの値０で指定される格納場所の値、即ち、配列Ｐ→Ｃの添字０の要素に、配列Ｃ→Ｐの添字１に対応するノード１のノード番号を設定する。これにより、ノード０の子ノードとしてノード１が設定される。このとき、配列Ａｇｇｒの中でノード０に対する要素の値をインクリメントする。これにより、ノード０の別の子ノードが検出されたとき、その子ノードの番号は、配列Ｐ→Ｃの添字１の要素の値として設定されることになる。本例では、配列Ｃ→Ｐを参照すると、ノード６の親ノードがノード０であるため、配列Ｐ→Ｃの添字１の要素の値として、このノード６のノード番号６が設定されている。
【０１４４】
次に、図４７の（Ｂ）手順５では、配列Ａｇｇｒを手順３の終了時の状態に戻す。これは、例えば、図４７の（Ｂ）に示されるように、配列Ａｇｇｒの要素を１個分ずつ後へシフトし、先頭に０を詰めることにより実現できる。或いは、手順３の終了時の配列Ａｇｇｒを別途保存しておいてもよい。或いは、配列Ａｇｇｒの先頭アドレスを一つ前に付け替てもよい。
【０１４５】
図４７の（Ｃ）には、変換によって得られた配列Ａｇｇｒと配列Ｐ→Ｃが示されている。これらの配列は、図１０に示した深さ優先「親→子」関係に基づく親子関係の配列と同じであるので、これ以上の説明は加えない。
【０１４６】
この変換方法では、既に説明したように、深さ優先又は幅優先の性質はそのまま保存される。したがって、本発明の一実施例による変換方法は、幅優先モードに基づく「子→親」表現から幅優先モードに基づく「親→子」表現への変換にも適合する。
【０１４７】
［「親→子」表現から「子→親」表現への変換］

次に、本発明の一実施例による親ノードに子ノードを対応付ける「親→子」関係から子ノードに親ノードを対応付ける「子→親」関係への変換方法を説明する。

【０１４８】

図４８は、本発明の一実施例による幅優先「子→親」表現から深さ優先「子→親」表現への変換方法のフローチャートである。親子関係は、親ノードに付与された番号の順に、当該親ノードに対応する子ノードに付与された番号を第１の配列の要素としてコンピュータシステム１０の記憶装置、例えば、ＲＡＭ１４に格納することにより定義されている。コンピュータシステム１０は、

子ノードに付与された番号の順に、当該子ノードに対応する親ノードに付与された番号を格納するため、前記記憶装置に第２の配列を確保するステップ４８０１と、

前記第１の配列の要素を順次に読み出し、当該要素と値が一致する番号が付与されたノードのために確保された前記第２の配列の要素として、前記第１の配列の要素に対する親ノードの番号を順次に格納するステップ４８０２と、

を実行する。これにより、親子関係は、「親→子」表現形式から「子→親」表現形式に変換される。即ち、変換後の親子関係は、子ノードに付与された番号の順に、当該子ノードに対応する親ノードに付与された番号を前記第２の配列の要素として前記記憶装置に格納することにより定義される。

【０１４９】

この変換方法では、深さ優先又は幅優先の性質はそのまま保存されるので、深さ優先モードに基づく「親→子」表現は、深さ優先モードに基づく「子→親」表現に変換され、幅優先モードに基づく「親→子」表現は、幅優先モードに基づく「子→親」表現に変換される。以下では、深さ優先モードに基づく「親→子」表現から深さ優先モードに基づく「子→親」表現への変換の例を説明する。本例では、図４５に示された深さ優先モードに基づくツリー型データ構造に関して、同図の（Ｃ）に示された「親→子」表現形式による親子関係を、同図の（Ｂ）に示された「子→親」表現形式による親子関係へ変換する。

【０１５０】

図４９は、本発明の一実施例による深さ優先モードに基づく「親→子」表現から深さ優先モードに基づく「子→親」表現への変換方法の説明図である。

【０１５１】

最初に、図４９の（Ａ）手順１に示されるように、配列Ｃ→Ｐの領域を確保し、－１で初期化する。

【０１５２】

次に、図４９の（Ｂ）手順２－１及び（Ｃ）手順２－２に示されるように、配列Ａｇｇｒと配列Ｐ→Ｃから「親→子」関係を読み出し、対応する配列Ｃ→Ｐの子ノード番号を埋める。例えば、配列Ａｇｇｒの添字０の要素（＝０）は、配列Ｐ→Ｃにおいて親ノード０の子ノードが格納されている領域の先頭を示し、配列Ａｇｇｒの添字１の要素（＝３）は、配列Ｐ→Ｃにおいて親ノード１の子ノードが格納されている領域の先頭を示しているので、親ノード０の子ノードのノード番号は、配列Ｐ→Ｃの添字０から添字２までの領域に格納されていることがわかる。この領域には、子ノード番号として、１、６及び８が順番に格納されているので、配列Ｃ→Ｐの添字１、添字６及び添字８の要素として、親ノード０のノード番号０を設定する。これにより、配列Ｃ→Ｐの中で、ノード番号０のノードを親ノードとする子ノードの領域が埋められる。この手続を配列Ａｇｇｒの添字の順に実行することにより、図４９の（Ｄ）に示される最終結果が得られる。

【０１５３】

この変換方法では、既に説明したように、深さ優先又は幅優先の性質はそのまま保存される。したがって、本発明の一実施例による変換方法は、幅優先モードに基づく「親→子」表現から幅優先モードに基づく「子→親」表現への変換にも適合する。

【０１５４】

［情報処理装置］

図５０は、本発明の一実施例によるツリー型データ構造を構築する情報処理装置５００

０のブロック図である。情報処理装置５０００は、ツリー型データ構造を表現するデータを記憶する記憶部５００１と、ルート・ノードを含むノードに固有のノード識別子を付与するノード定義部５００２と、前記ルート・ノード以外のノードである非ルート・ノードの各々に付与されたノード識別子に、前記非ルート・ノードの各々の親ノードに付与されたノード識別子を関連付ける親子関係定義部５００３と、を含む。

【０１５５】

好ましくは、ノード定義部５００２は、ノード識別子として数値を用い、より好ましくは、ノード識別子として連続する整数を用いる。

【０１５６】

また、親子関係定義部５００３は、非ルート・ノードの各々に付与されたノード識別子と、関連付けられた親ノードに付与されたノード識別子と、の組の配列を記憶部５００１に格納する。

【０１５７】

本発明は、以上の実施の形態に限定されることなく、特許請求の範囲に記載された発明の範囲内で、種々の変更が可能であり、それらも本発明の範囲内に包含されるものであることは言うまでもない。

【図面の簡単な説明】

【０１５８】

【図１】本発明の実施の形態にかかるツリー型データ構造を取り扱うコンピュータシステムのブロックダイヤグラムである。

【図２】ツリー形式データの一例であるＰＯＳデータの説明図であり、（Ａ）は、このツリー形式データのデータ構造（即ち、トポロジー）及びデータ値を視覚的に表現した例であり、（Ｂ）は、同じツリー形式データをＸＭＬ形式で表現した例である。

【図３】アークリストを用いたツリー型データ構造の表現形式の一例の説明図である。

【図４】本発明の一実施例による「子→親」関係に基づくツリー型データ構造の表現方法の説明図である。

【図５】本発明の一実施例によるツリー型データ構造を記憶装置上に構築する方法のフローチャートである。

【図６】本発明の一実施例によりＩＤ形式のツリー構造型データを整数連番形式のツリー構造型データへ変換する処理の説明図である。

【図７】本発明の他の一実施例によりＩＤ形式のツリー構造型データを整数連番形式のツリー構造型データへ変換する処理の説明図である。

【図８】本発明の一実施例による深さ優先に基づくノード定義処理のフローチャートである。

【図９】本発明の一実施例により作成された「子→親」表現に基づく親子関係の配列の説明図である。

【図１０】図６に示された深さ優先のツリー型データ構造から作成された「親→子」表現に基づく親子関係の配列の説明図である。

【図１１】本発明の一実施例による幅優先に基づくノード定義処理のフローチャートである。

【図１２】本発明の一実施例により作成された「子→親」表現に基づく親子関係の配列の説明図である。

【図１３】図７に示された深さ優先のツリー型データ構造から作成された「親→子」表現に基づく親子関係の配列の説明図である。

【図１４】本発明の一実施例による三つの表現形式の相互変換の関係を示す図である。

【図１５】本発明の一実施例によるコンピュータシステムによって実現されるツリー型データ構造の構築方法のフローチャートである。

【図１６】本発明の一実施例による深さ優先「子→親」表現から幅優先「子→親」表

現への変換の説明図である。
【図１７】本発明の一実施例による深さ優先「子→親」表現から幅優先「子→親」表現への変換方法のフローチャートである。
【図１８】本発明の一実施例による深さ優先に基づくツリー型データ構造の世代毎に属しているノードの個数を計数する処理の説明図である。
【図１９】本発明の一実施例による深さ優先に基づくツリー型データ構造の世代毎に属しているノードの個数を計数する処理（手順０～２）の説明図である。
【図２０】本発明の一実施例による深さ優先に基づくツリー型データ構造の世代毎に属しているノードの個数を計数する処理（手順３～５）の説明図である。
【図２１】本発明の一実施例による深さ優先に基づくツリー型データ構造の世代毎に属しているノードの個数を計数する処理（手順６～８）の説明図である。
【図２２】本発明の一実施例による深さ優先に基づくツリー型データ構造の世代毎に属しているノードの個数を計数する処理（手順９～１１）の説明図である。
【図２３】本発明の一実施例による深さ優先に基づくツリー型データ構造の世代毎に属しているノードの個数を計数する処理（手順１２）の説明図である。
【図２４】本発明の一実施例によるノードの番号を幅優先モードで付与される番号に変換する変換配列を作成する処理（手順０～２）の説明図である。
【図２５】本発明の一実施例によるノードの番号を幅優先モードで付与される番号に変換する変換配列を作成する処理（手順３～５）の説明図である。
【図２６】本発明の一実施例によるノードの番号を幅優先モードで付与される番号に変換する変換配列を作成する処理（手順６～８）の説明図である。
【図２７】本発明の一実施例によるノードの番号を幅優先モードで付与される番号に変換する変換配列を作成する処理（手順９～１１）の説明図である。
【図２８】本発明の一実施例によるノードの番号を幅優先モードで付与される番号に変換する変換配列を作成する処理（手順１２）の説明図である。
【図２９】本発明の一実施例による深さ優先に基づくノードの親子関係を幅優先に基づくノードの親子関係に変換する処理の説明図である。
【図３０】本発明の一実施例による幅優先「子→親」表現から深さ優先「子→親」表現への変換の説明図である。
【図３１】本発明の一実施例による幅優先「子→親」表現から深さ優先「子→親」表現への変換方法のフローチャートである。
【図３２】本発明の一実施例による幅優先に基づくツリー型データ構造の各ノードの子孫の個数を計数する処理の説明図（その１）である
【図３３】本発明の一実施例による幅優先に基づくツリー型データ構造の各ノードの子孫の個数を計数する処理の説明図（その２）である
【図３４】本発明の一実施例による幅優先モードの番号から深さ優先モードの番号への変換配列を作成する処理の説明図である。
【図３５】本発明の一実施例による幅優先に基づくノードの親子関係を深さ優先に基づくノードの親子関係に変換する処理の説明図である。
【図３６】本発明の他の一実施例による幅優先「子→親」表現から深さ優先「子→親」表現への変換方法の説明図（その１）である。
【図３７】本発明の他の一実施例による幅優先「子→親」表現から深さ優先「子→親」表現への変換方法の説明図（その２）である。
【図３８】本発明の他の一実施例による幅優先「子→親」表現から深さ優先「子→親」表現への変換方法の説明図（その３）である。
【図３９】本発明の他の一実施例による幅優先「子→親」表現から深さ優先「子→親」表現への変換方法の説明図（その４）である。
【図４０】本発明の他の一実施例による幅優先「子→親」表現から深さ優先「子→親」表現への変換方法の説明図（その５）である。
【図４１】本発明の他の一実施例による幅優先「子→親」表現から深さ優先「子→親

」表現への変換方法の説明図（その６）である。
【図４２】本発明の他の一実施例による幅優先「子→親」表現から深さ優先「子→親」表現への変換方法の説明図（その７）である。
【図４３】本発明の他の一実施例による幅優先「子→親」表現から深さ優先「子→親」表現への変換方法の説明図（その８）である。
【図４４】本発明の一実施例による「子→親」表現から「親→子」表現への変換方法のフローチャートである。
【図４５】深さ優先モードに基づくツリー型データ構造の一例の説明図である。
【図４６】本発明の一実施例による深さ優先モードに基づく「子→親」表現から深さ優先モードに基づく「親→子」表現への変換方法の説明図（その１）である。
【図４７】本発明の一実施例による深さ優先モードに基づく「子→親」表現から深さ優先モードに基づく「親→子」表現への変換方法の説明図（その２）である。
【図４８】本発明の一実施例による「親→子」表現から「子→親」表現への変換方法のフローチャートである。
【図４９】本発明の一実施例による深さ優先モードに基づく「親→子」表現から深さ優先モードに基づく「子→親」表現への変換方法の説明図である。
【図５０】本発明の一実施例によるツリー型データ構造を記憶装置上に構築する情報処理装置のブロック図である。

【符号の説明】
【０１５９】

１０	コンピュータシステム
１２	ＣＰＵ
１４	ＲＡＭ
１６	ＲＯＭ
１８	固定記憶装置
２０	ＣＤ－ＲＯＭドライバ
２２	Ｉ／Ｆ
２４	入力装置
２６	表示装置
５０００	情報処理装置
５００１	記憶部
５００２	ノード定義部
５００３	親子関係定義部

【書類名】図面
【図１】

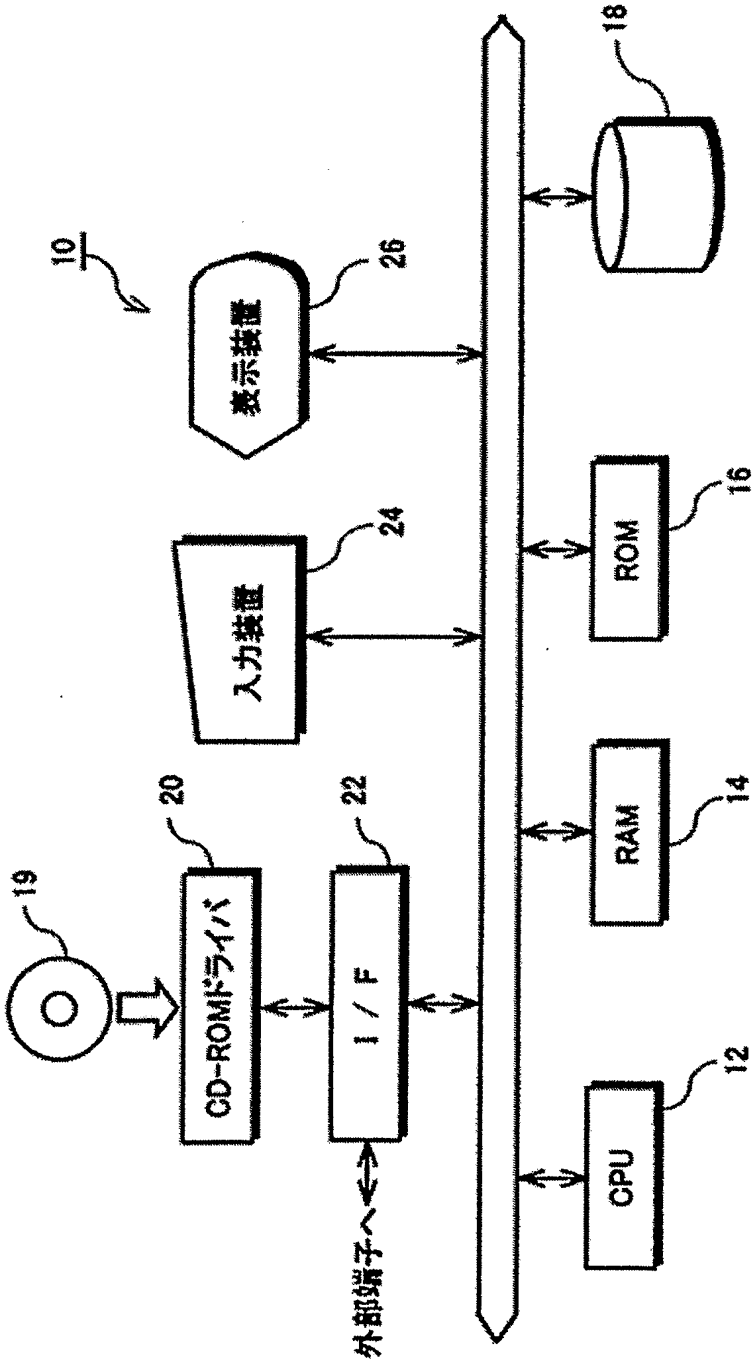
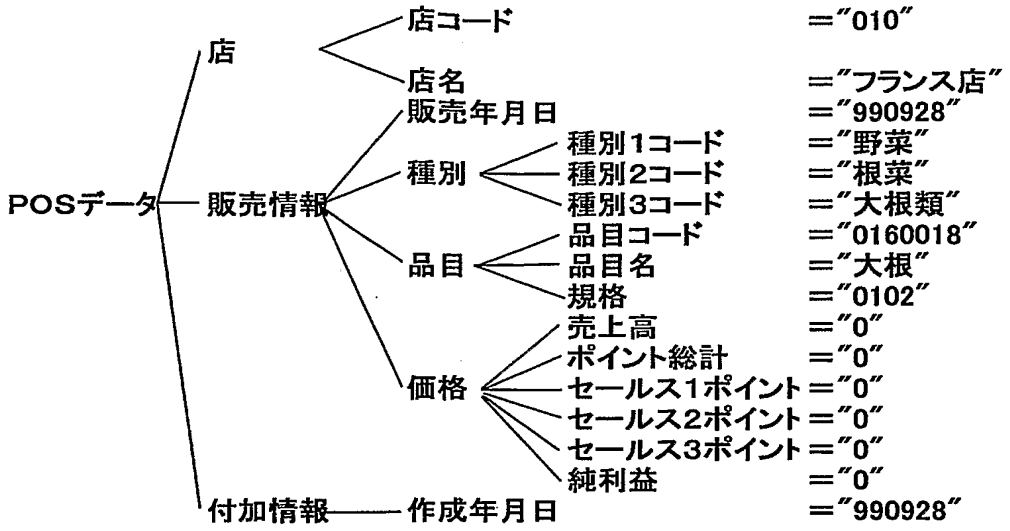

10
表示装置
26
18
入力装置
24
ROM
16
RAM
14
20
CD-ROMドライバ
19
22
I / F
外部端子へ
CPU
12

図1

【図２】

図2

(A)



(B)

```
<posdata>
  <shop>
    <shopCode>010</shopCode>
    <shopName>フランス店</shopName>
  </shop>
  <salesInformation>
    <sellDate>990928</sellDate>
    <class>
      <class1 code="01">野菜</class1>
      <class2 code="01">根菜</class2>
      <class3 code="01">大根類</class3>
    </class>
    <goods>
      <goodsCode>"0160018"</goodsCode>
      <goodsName>大根</goodsName>
      <standard>0102</standard>
    </goods>
    <price>
      <amountOfSales>0</amountOfSales>
      <amountOfPoints>0</amountOfPoints>
      <sales1 point="0">0</sales1>
      <sales2 point="0">0</sales2>
      <sales3 point="0">0</sales3>
      <grossProfit>0</grossProfit>
    </price>
  </salesInformation>
  <additionalInformation>
    <createdDate>990928</createdDate>
  </additionalInformation>
</posdata>
```

【図３】

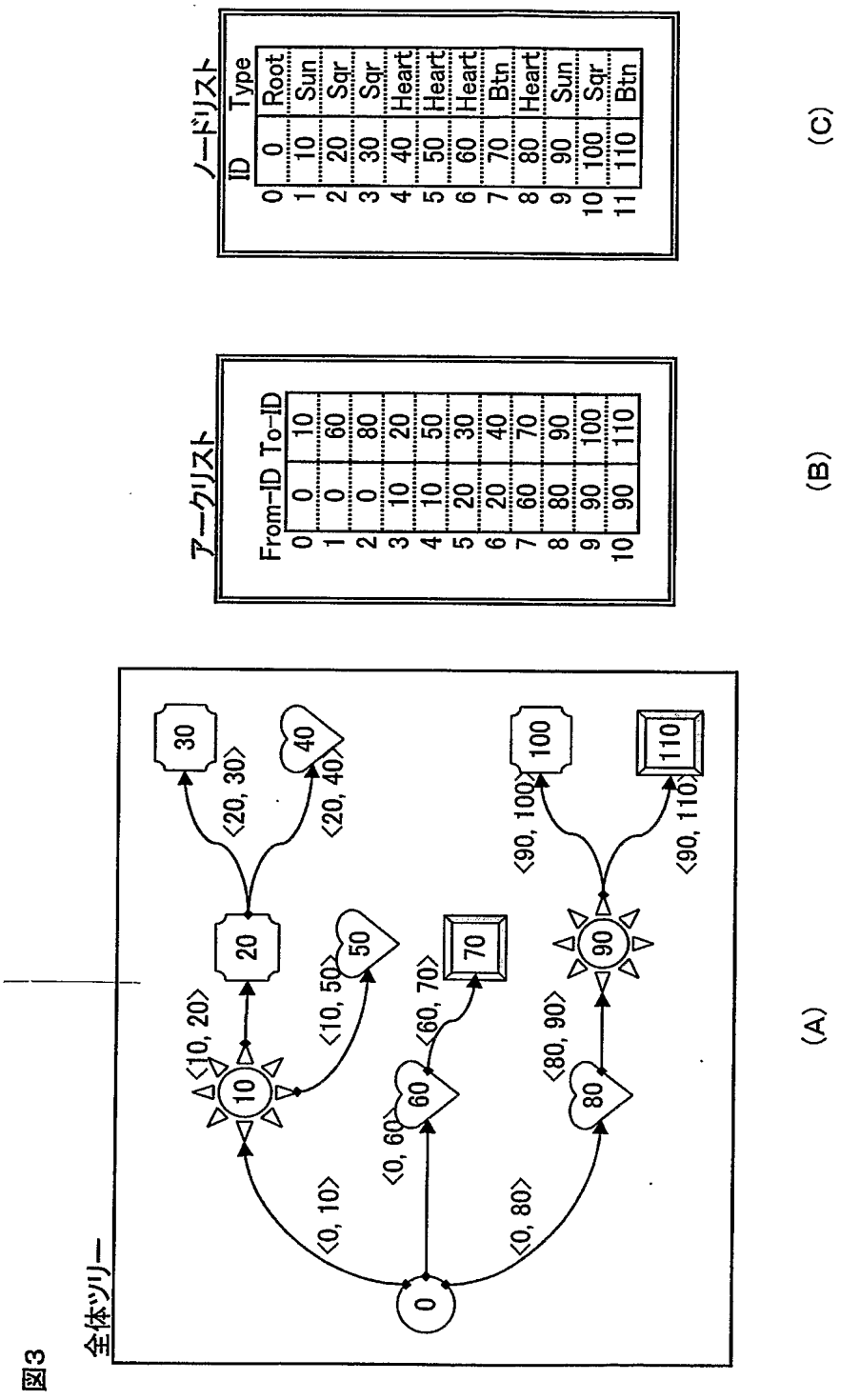

全体ツリー
0
10
20
30
40
50
60
70
80
90
100
110
<0, 10>
<10, 20>
<20, 30>
<20, 40>
<10, 50>
<0, 60>
<60, 70>
<0, 80>
<80, 90>
<90, 100>
<90, 110>
(A)
アークリスト
From-ID To-ID
0 0 10
1 0 60
2 0 80
3 10 20
4 10 50
5 20 30
6 20 40
7 60 70
8 80 90
9 90 100
10 90 110
(B)
ノードリスト
ID Type
0 0 Root
1 10 Sun
2 20 Sqr
3 30 Sqr
4 40 Heart
5 50 Heart
6 60 Heart
7 70 Btn
8 80 Heart
9 90 Sun
10 100 Sqr
11 110 Btn
(C)

図3

【図４】

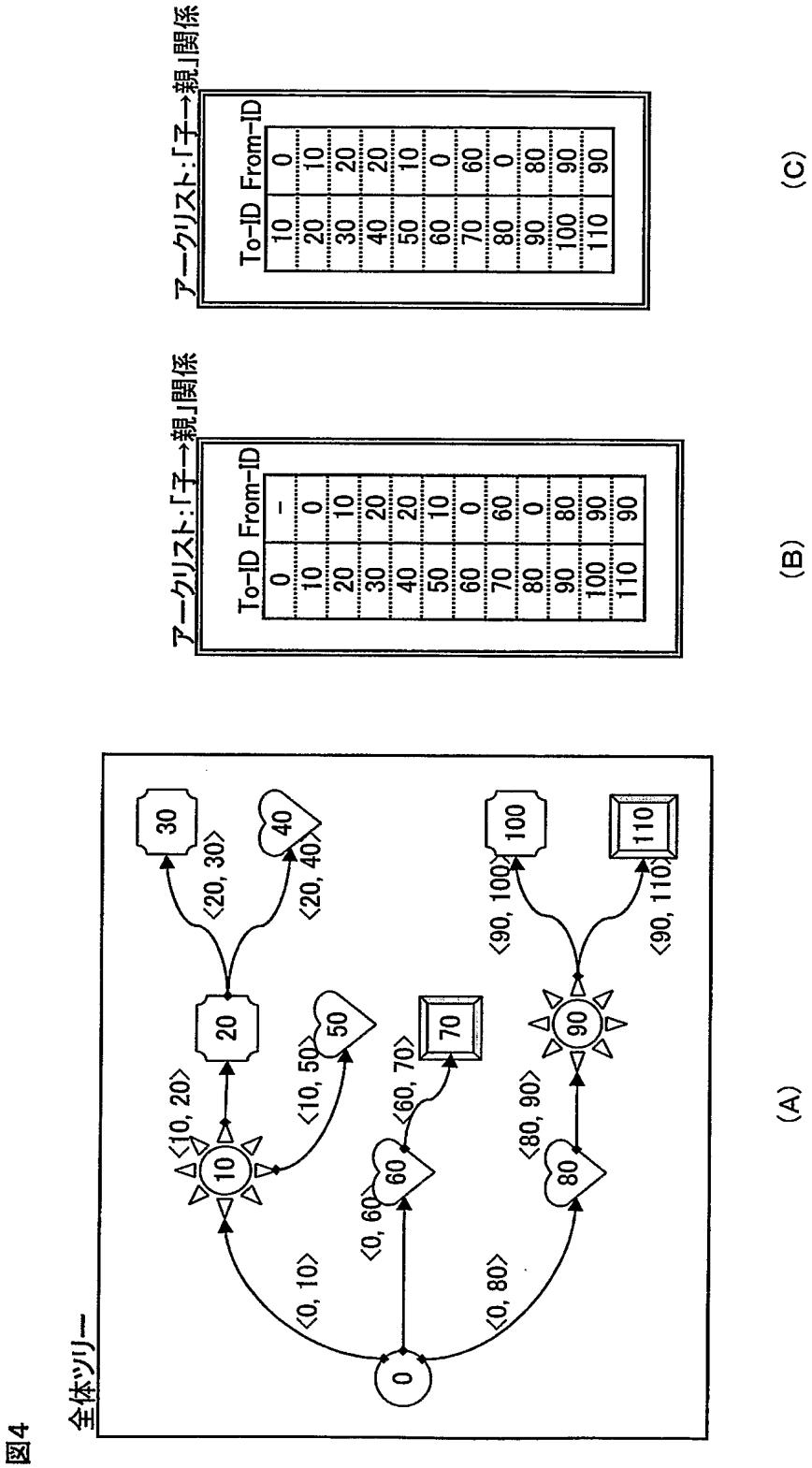
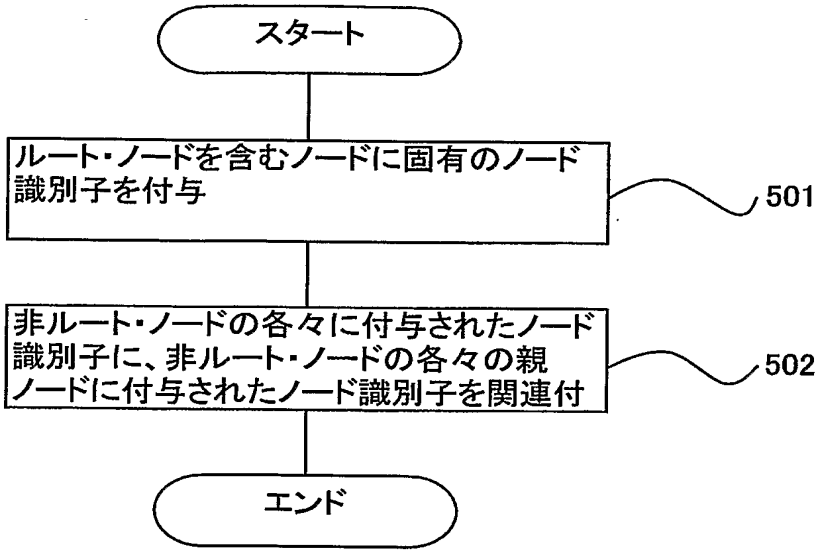


図4
全体ツリー
0
10
20
30
40
50
60
70
80
90
100
110
<0, 10>
<10, 20>
<20, 30>
<20, 40>
<10, 50>
<0, 60>
<60, 70>
<0, 80>
<80, 90>
<90, 100>
<90, 110>
(A)
アークリスト:「子→親」関係
To-ID From-ID
0 –
10 0
20 10
30 20
40 20
50 10
60 0
70 60
80 0
90 80
100 90
110 90
(B)
アークリスト:「子→親」関係
To-ID From-ID
10 0
20 10
30 20
40 20
50 10
60 0
70 60
80 0
90 80
100 90
110 90
(C)

【図５】

図5


スタート
ルート・ノードを含むノードに固有のノード識別子を付与
501
非ルート・ノードの各々に付与されたノード識別子に、非ルート・ノードの各々の親ノードに付与されたノード識別子を関連付
502
エンド

【図６】

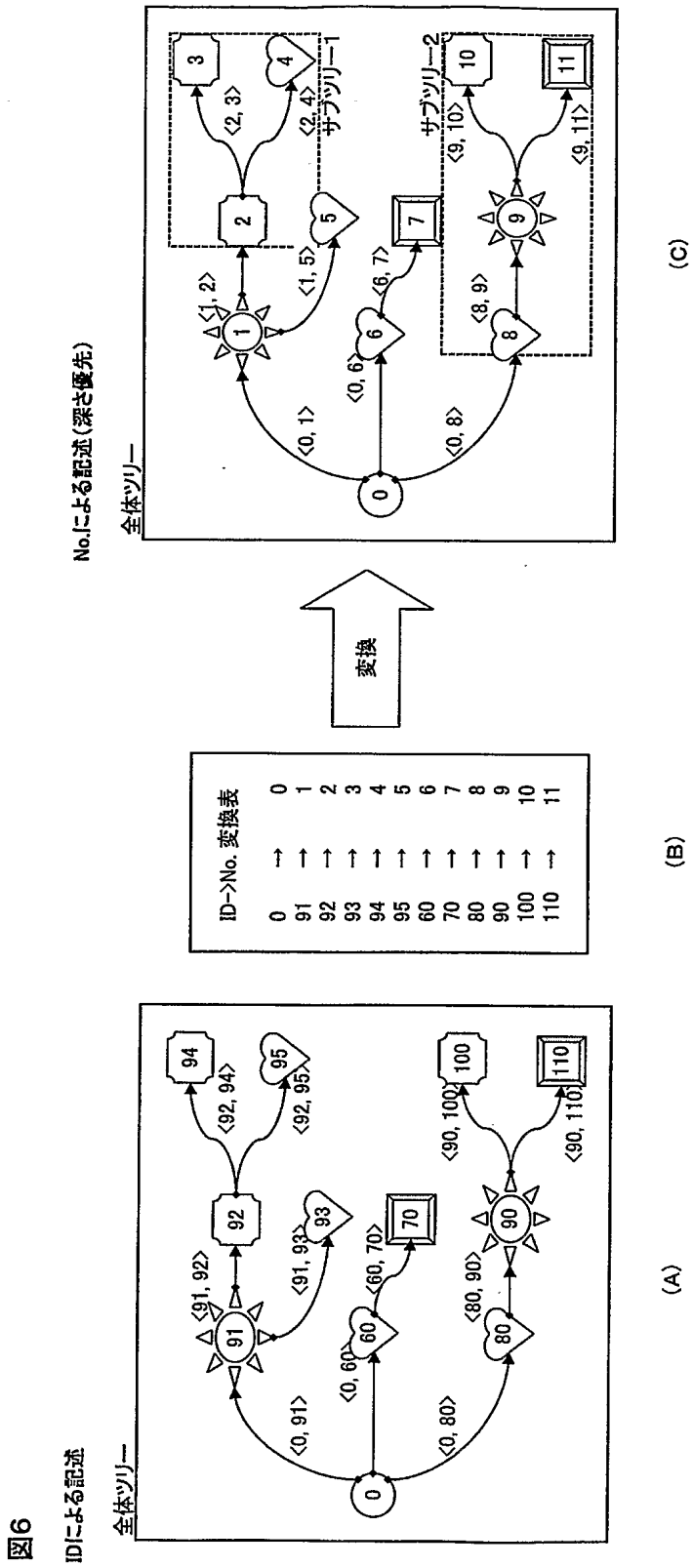


図6
IDによる記述
全体ツリー
(A)
ID->No. 変換表
0 0
91 1
92 2
93 3
94 4
95 5
60 6
70 7
80 8
90 9
100 10
110 11
(B)
変換
No.による記述(深さ優先)
全体ツリー
サブツリー1
サブツリー2
(C)

【図７】

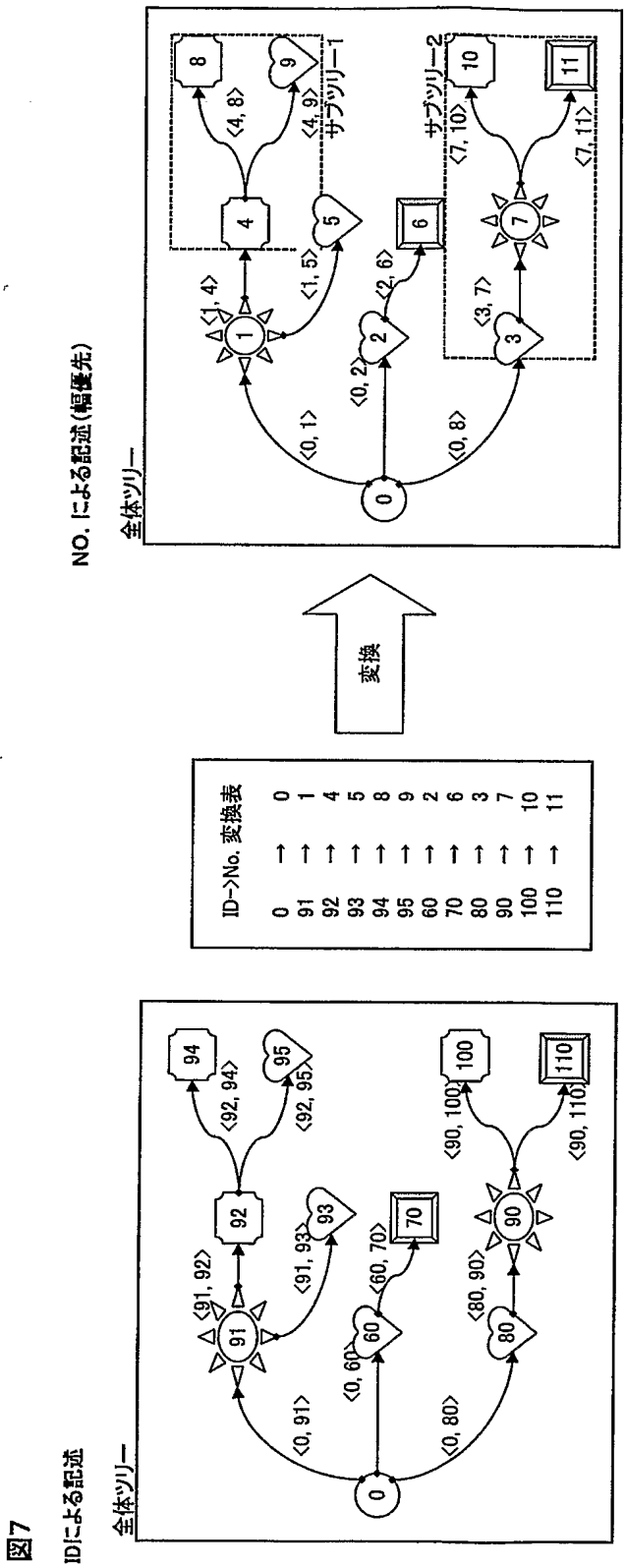
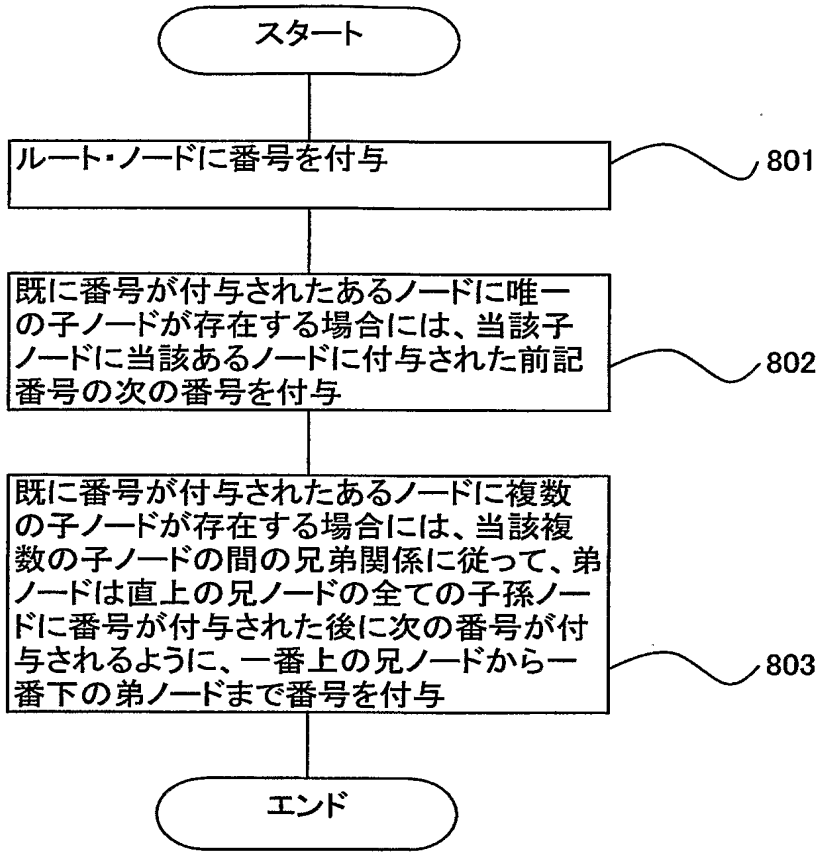


図7
IDによる記述
全体ツリー
<0, 91>
<0, 60>
<0, 80>
<91, 92>
<91, 93>
<60, 70>
<80, 90>
<92, 94>
<92, 95>
<90, 100>
<90, 110>
変換
ID->No. 変換表
0 → 0
91 → 1
92 → 4
93 → 5
94 → 8
95 → 9
60 → 2
70 → 6
80 → 3
90 → 7
100 → 10
110 → 11
NO. による記述(幅優先)
全体ツリー
<0, 1>
<0, 2>
<0, 8>
<1, 4>
<1, 5>
<2, 6>
<3, 7>
<4, 8>
<4, 9>
<7, 10>
<7, 11>
サブツリー1
サブツリー2

【図８】

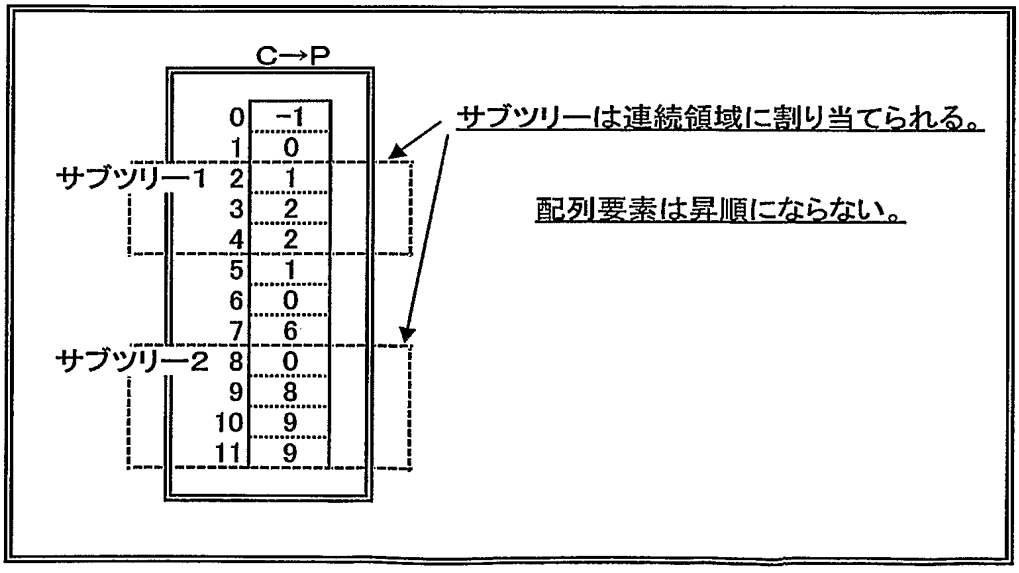
図8



【図９】

図9

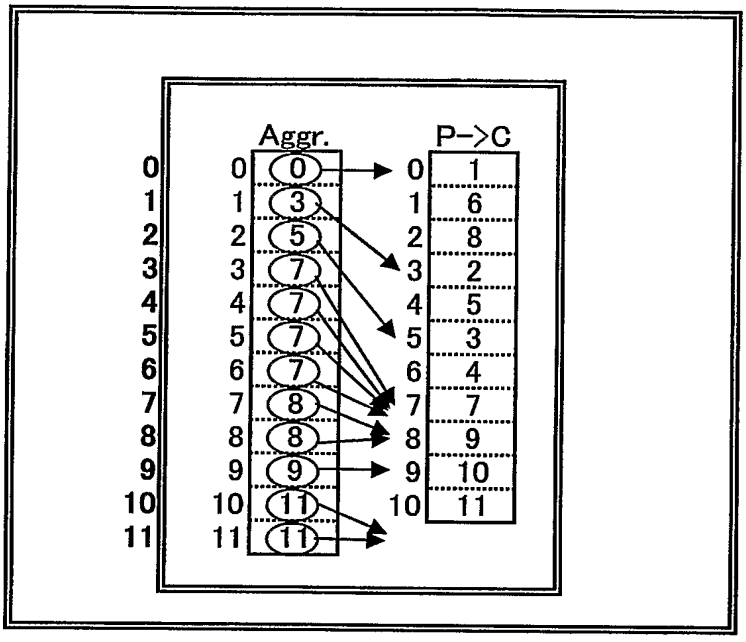
深さ優先「子→親」関係に基づく親子関係の配列



【図１０】

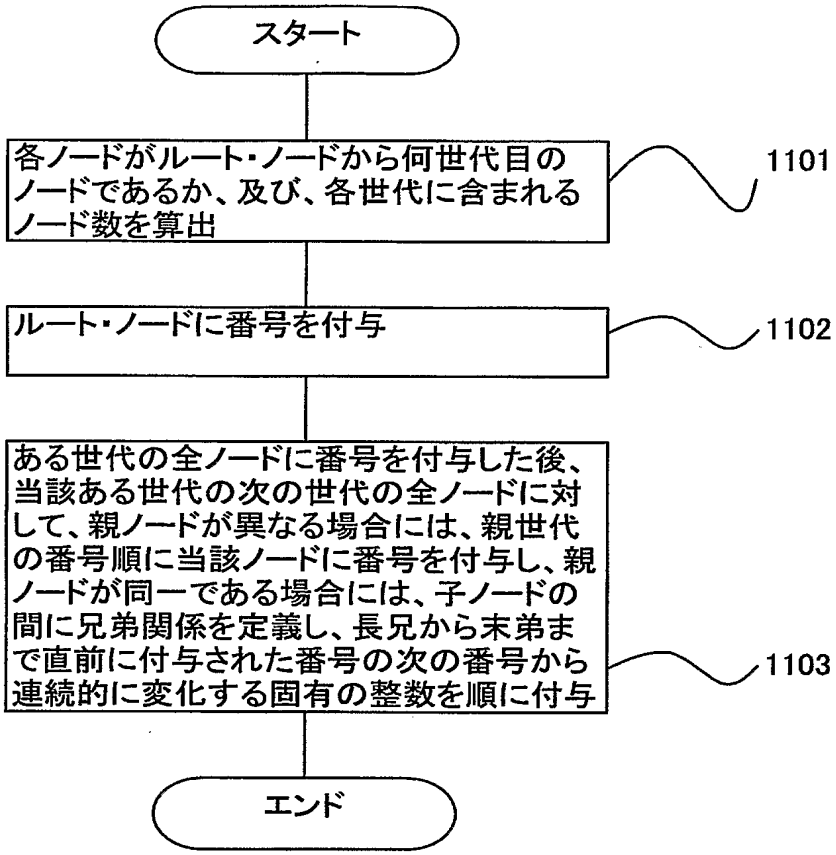
図10

深さ優先「親→子」関係に基づく親子関係の配列



【図１１】

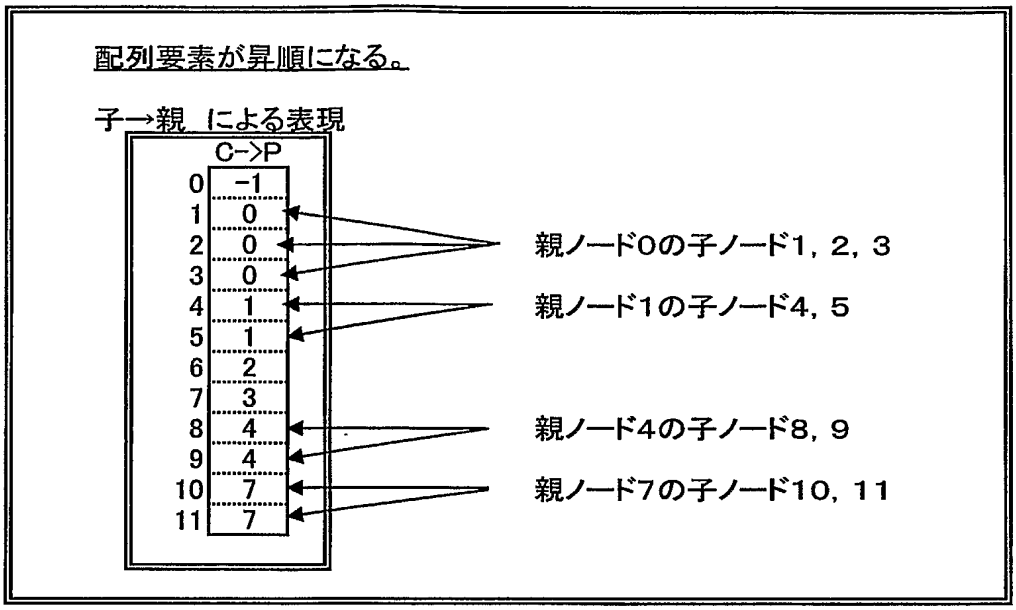
図11


スタート
各ノードがルート・ノードから何世代目のノードであるか、及び、各世代に含まれるノード数を算出
1101
ルート・ノードに番号を付与
1102
ある世代の全ノードに番号を付与した後、当該ある世代の次の世代の全ノードに対して、親ノードが異なる場合には、親世代の番号順に当該ノードに番号を付与し、親ノードが同一である場合には、子ノードの間に兄弟関係を定義し、長兄から末弟まで直前に付与された番号の次の番号から連続的に変化する固有の整数を順に付与
1103
エンド

【図１２】

図12

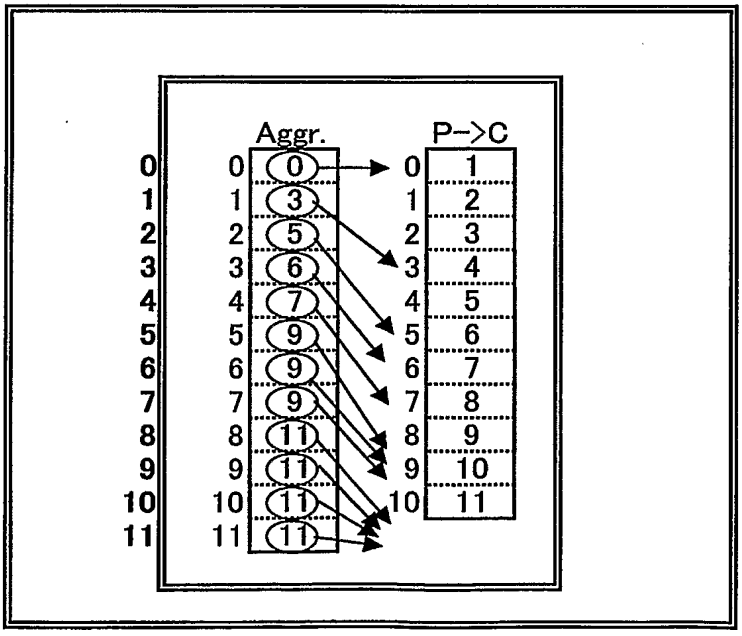
幅優先「子→親」関係に基づく親子関係の配列



【図１３】

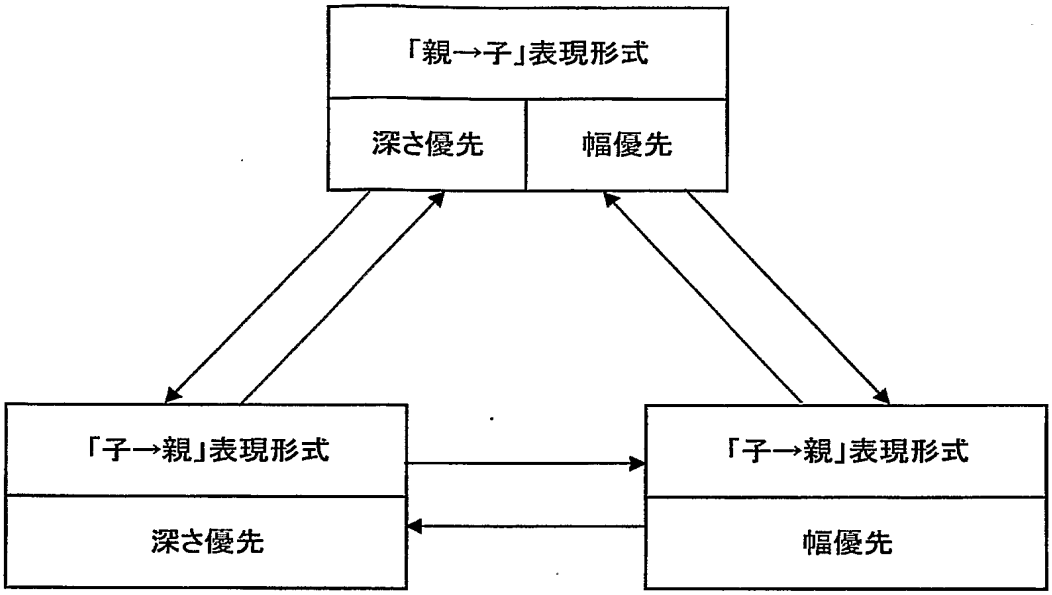
図13

幅優先「親→子」関係に基づく親子関係の配列



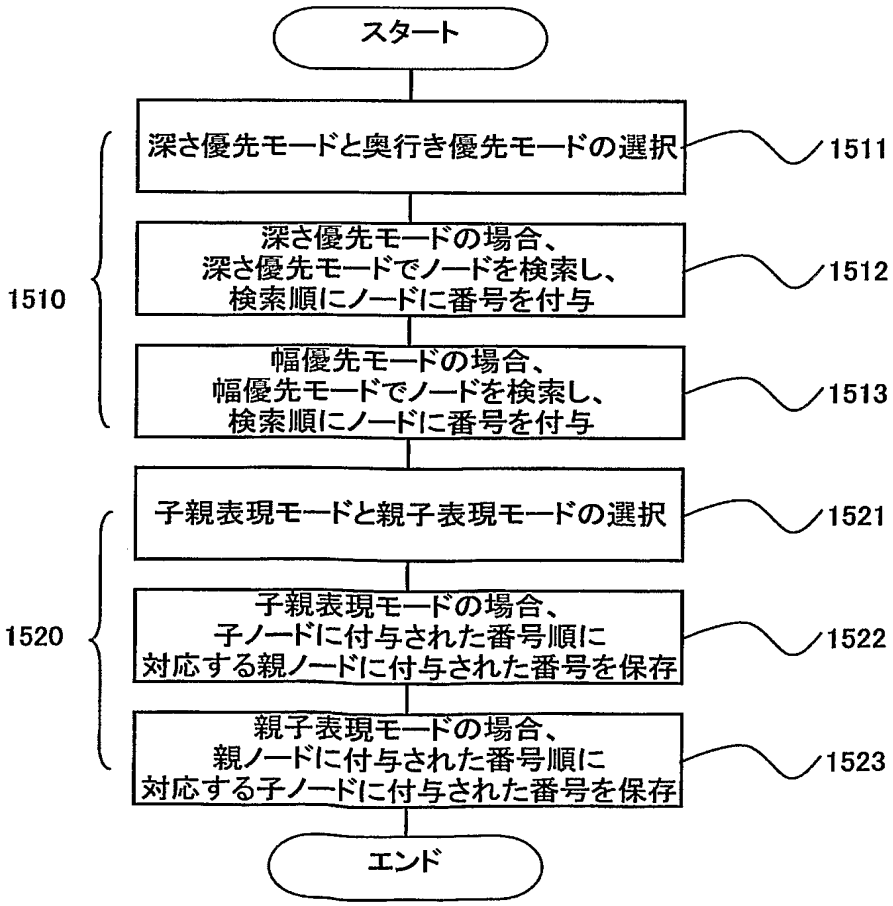
【図１４】

図14



【図１５】

図15



【図１６】

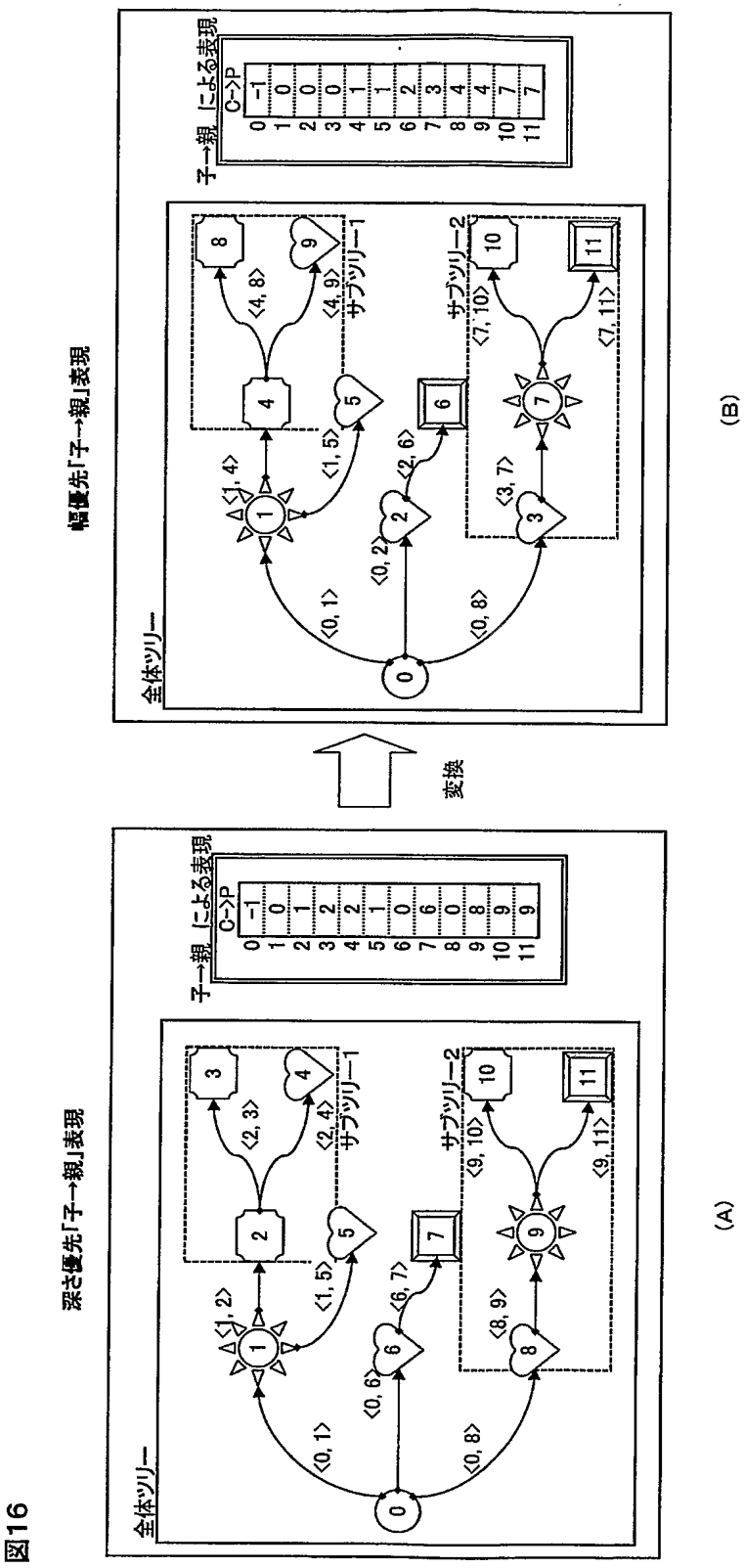
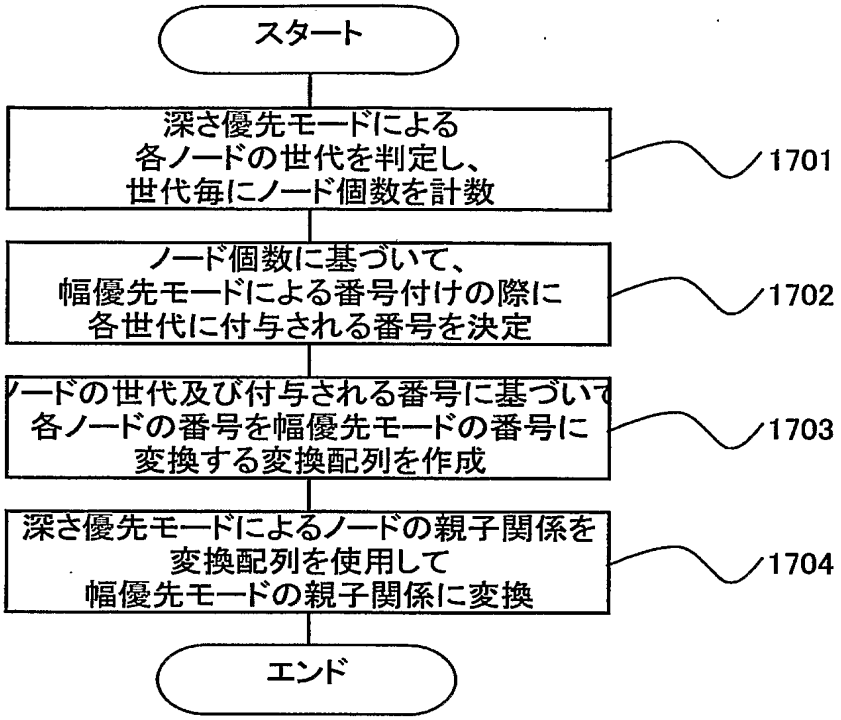


図16
深さ優先「子→親」表現
全体ツリー
子→親 による表現
C->P
サブツリー1
サブツリー2
(A)
変換
幅優先「子→親」表現
全体ツリー
子→親 による表現
C->P
サブツリー1
サブツリー2
(B)

【図１７】

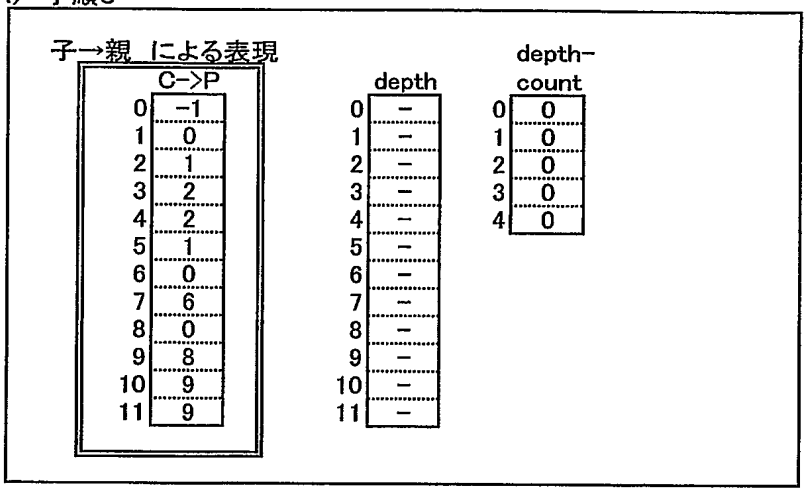
図17


スタート
深さ優先モードによる
各ノードの世代を判定し、
世代毎にノード個数を計数
1701
ノード個数に基づいて、
幅優先モードによる番号付けの際に
各世代に付与される番号を決定
1702
ノードの世代及び付与される番号に基づいて
各ノードの番号を幅優先モードの番号に
変換する変換配列を作成
1703
深さ優先モードによるノードの親子関係を
変換配列を使用して
幅優先モードの親子関係に変換
1704
エンド

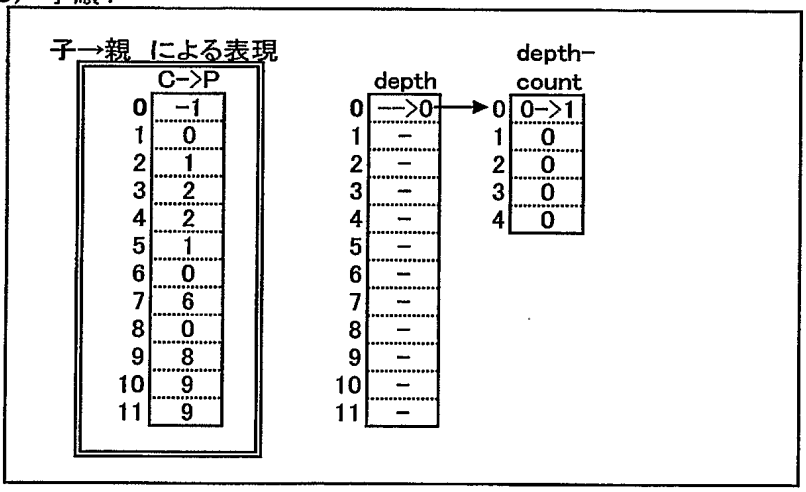
【図１８】

図18

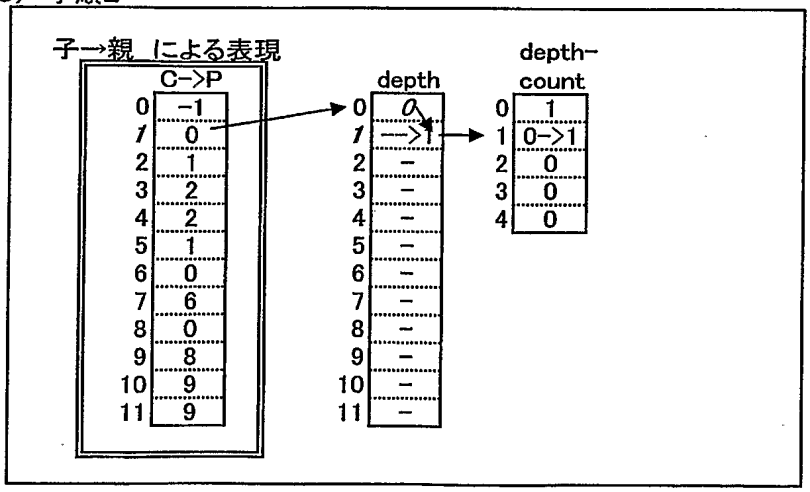
(A) 手順0



(B) 手順1



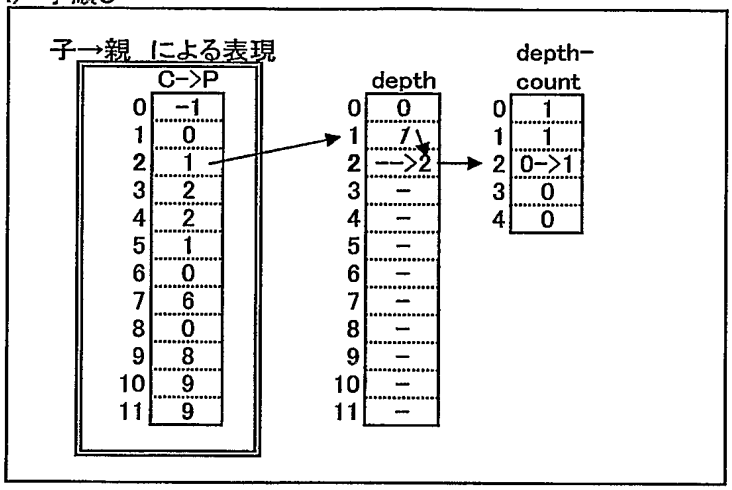
(C) 手順2



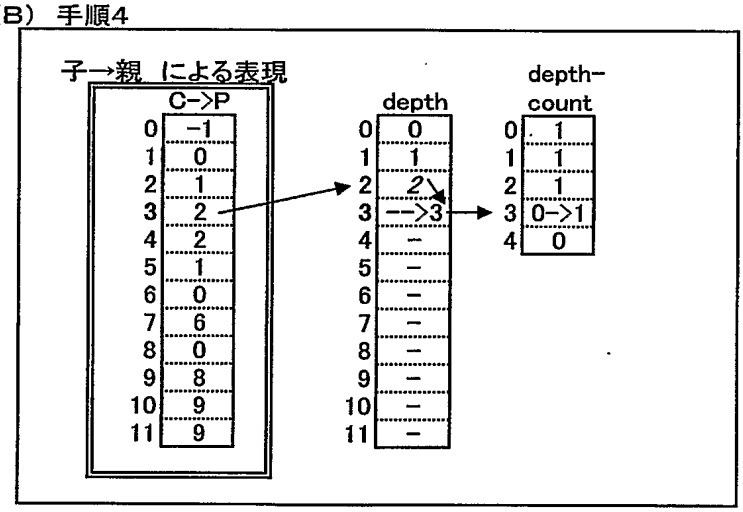
【図１９】

図19

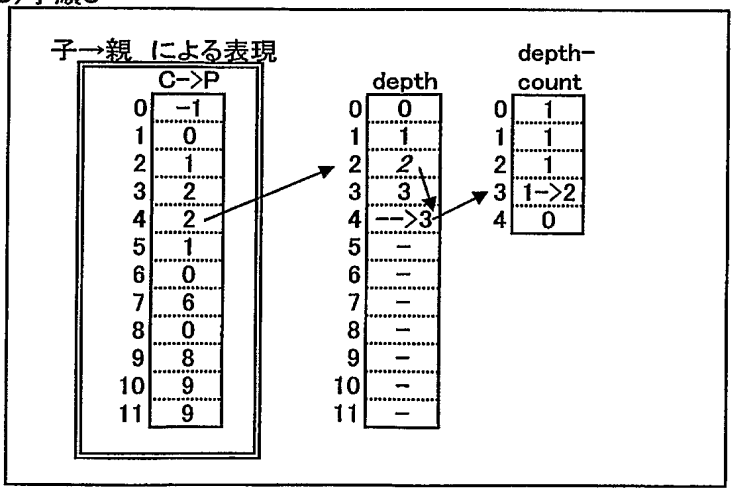
(A)　手順3



(B)　手順4



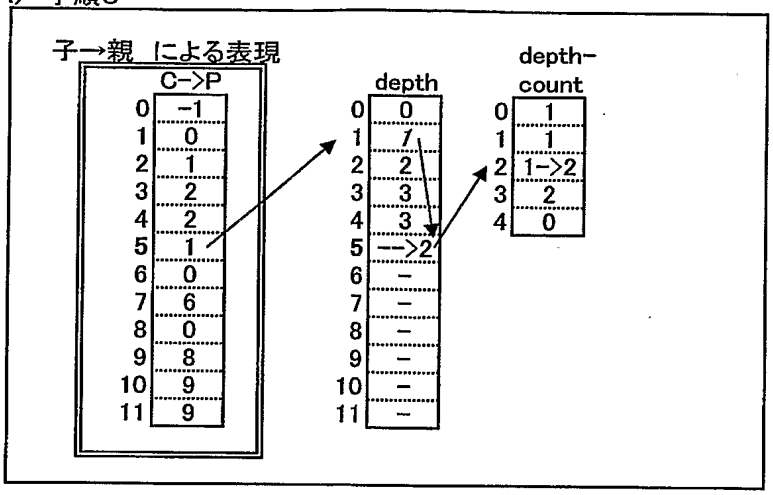
(C)手順5



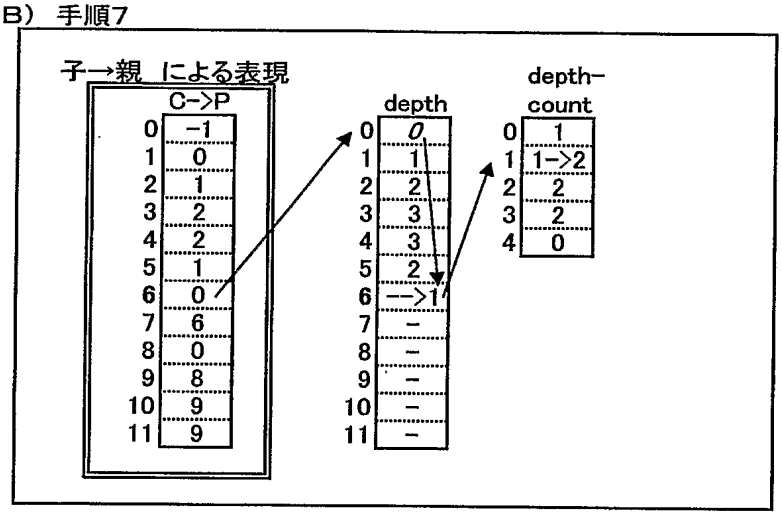
【図２０】

図20

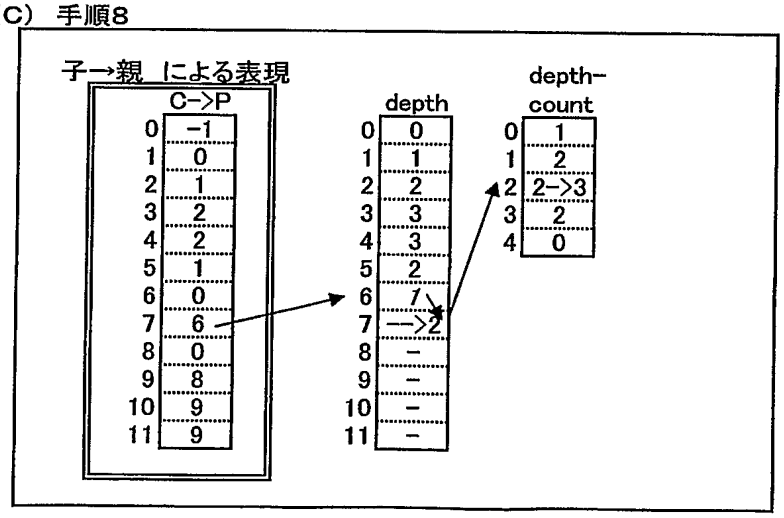
(A)　手順6



(B)　手順7



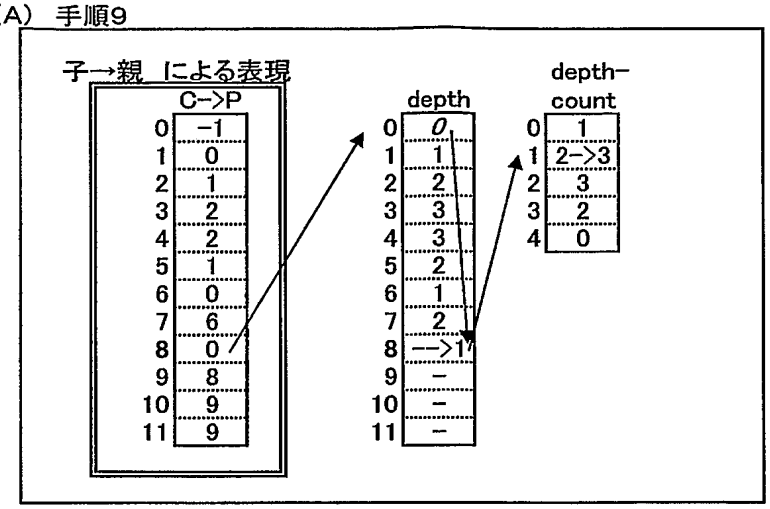
(C)　手順8



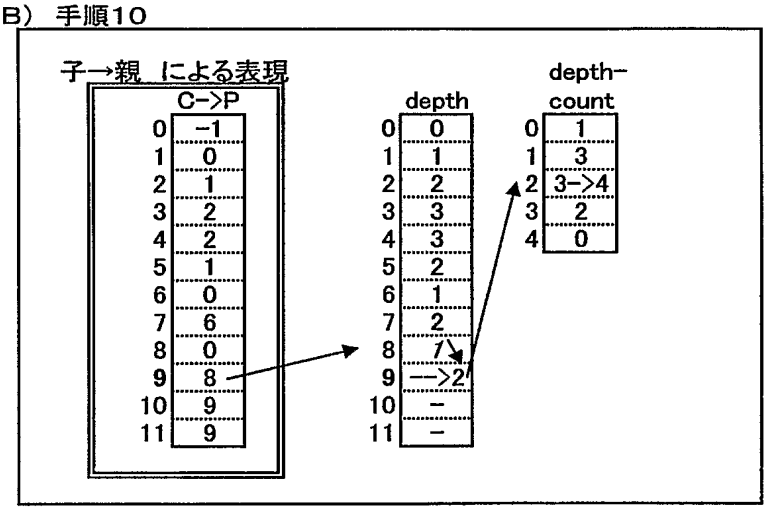
【図２１】

図21

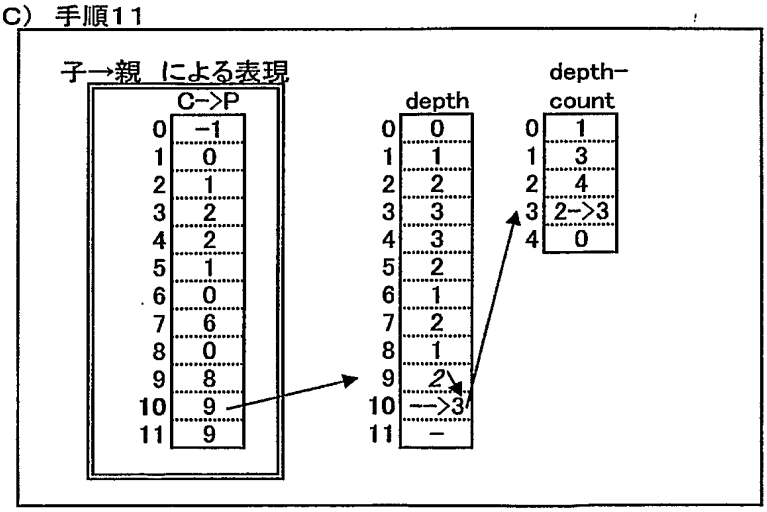
(A) 手順9



(B) 手順10

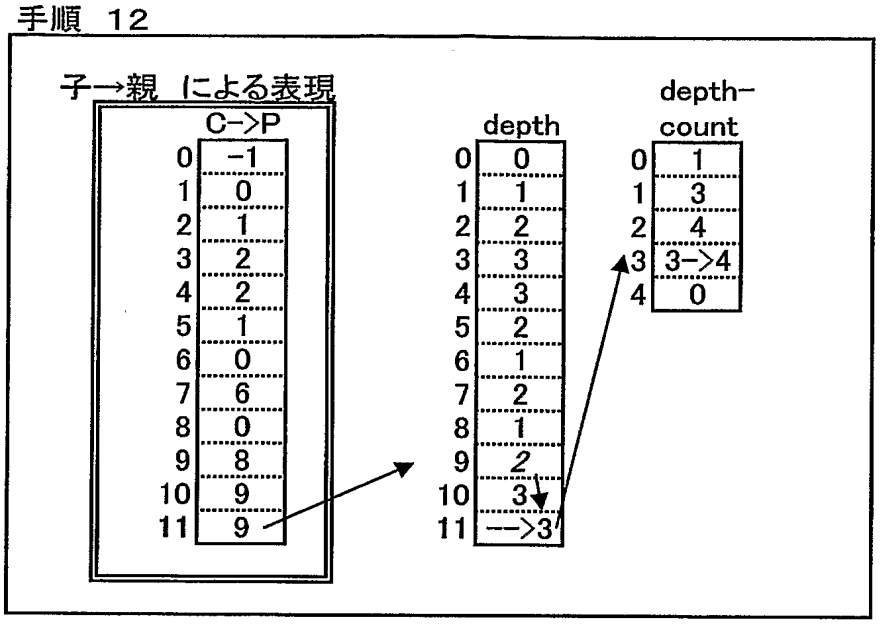


(C) 手順11



【図２２】

図22


手順　12
子→親　による表現
C->P
0 -1
1 0
2 1
3 2
4 2
5 1
6 0
7 6
8 0
9 8
10 9
11 9
depth
0 0
1 1
2 2
3 3
4 3
5 2
6 1
7 2
8 1
9 2
10 3
11 —>3
depth-
count
0 1
1 3
2 4
3 3->4
4 0

【図２３】

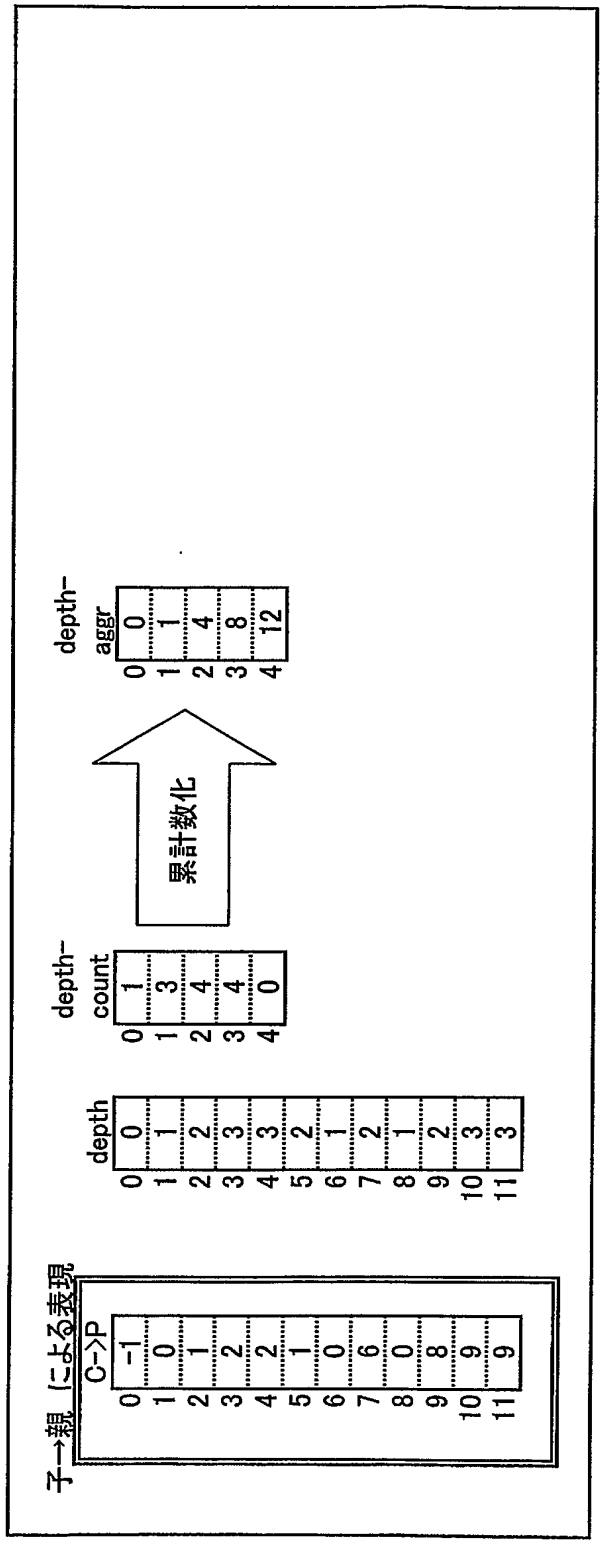
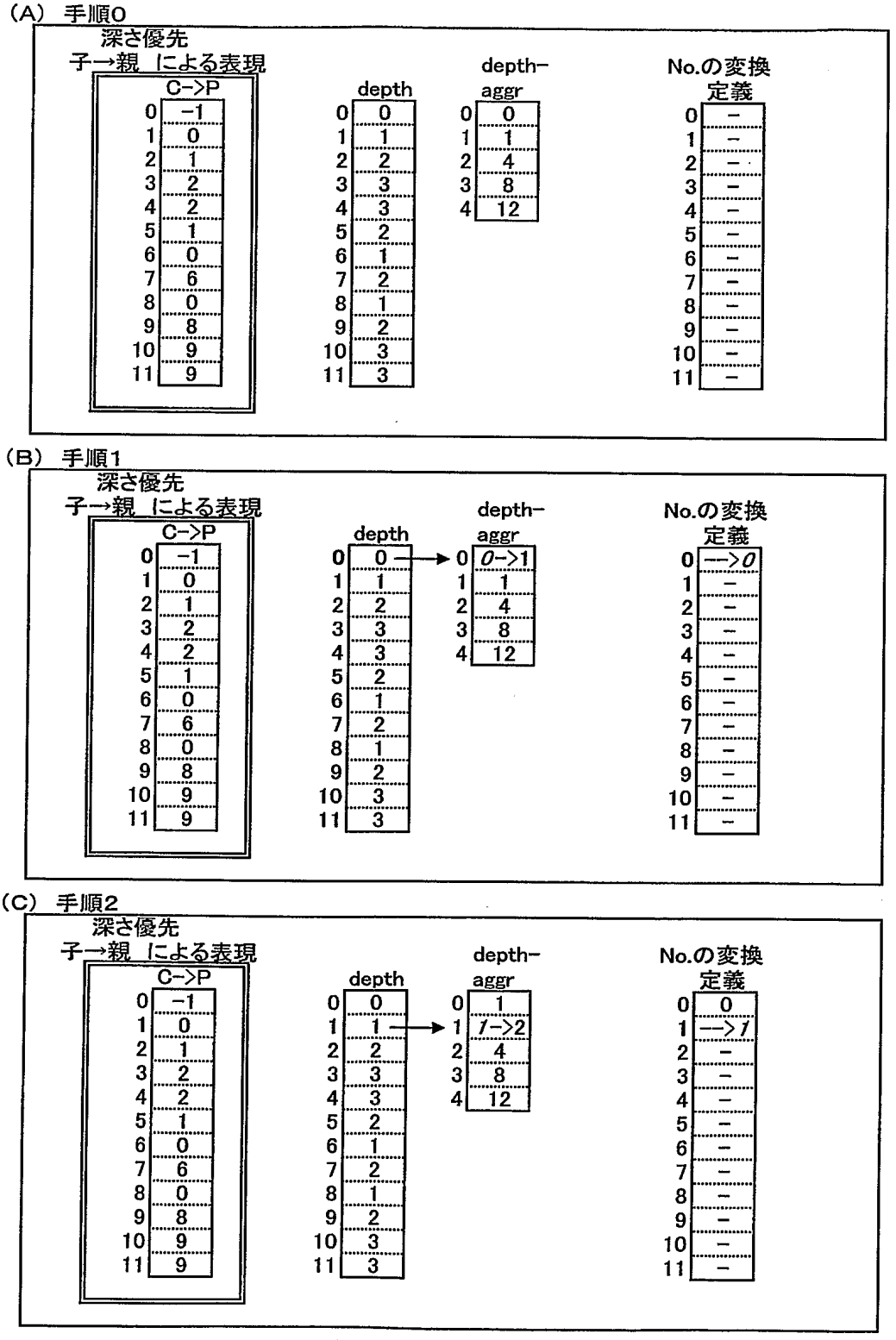


図23

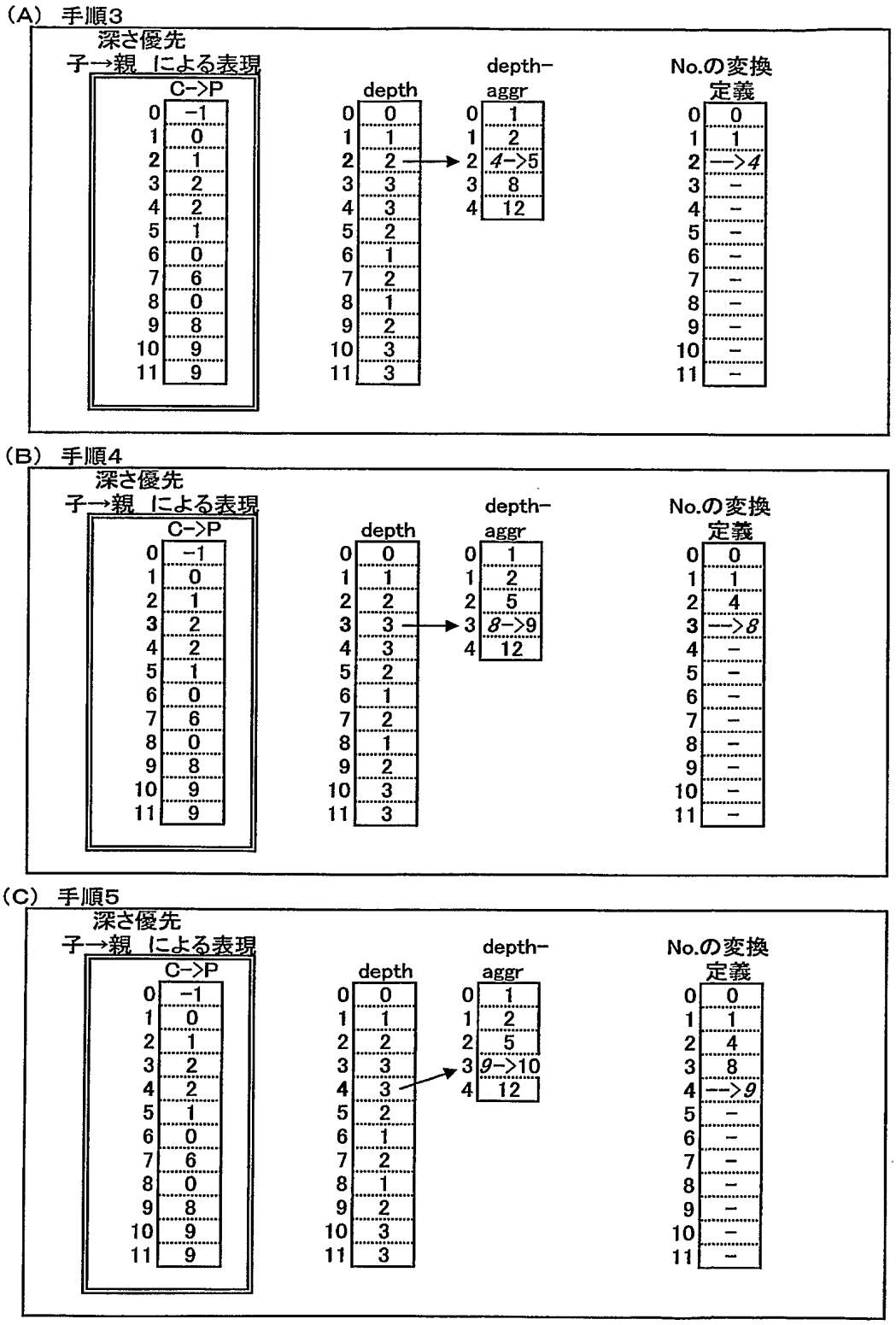
【図２４】

図24


(A) 手順0
深さ優先
子→親 による表現
C->P
depth
depth-aggr
No.の変換定義
(B) 手順1
深さ優先
子→親 による表現
C->P
depth
depth-aggr
No.の変換定義
0->1
--->0
(C) 手順2
深さ優先
子→親 による表現
C->P
depth
depth-aggr
No.の変換定義
1->2
--->1

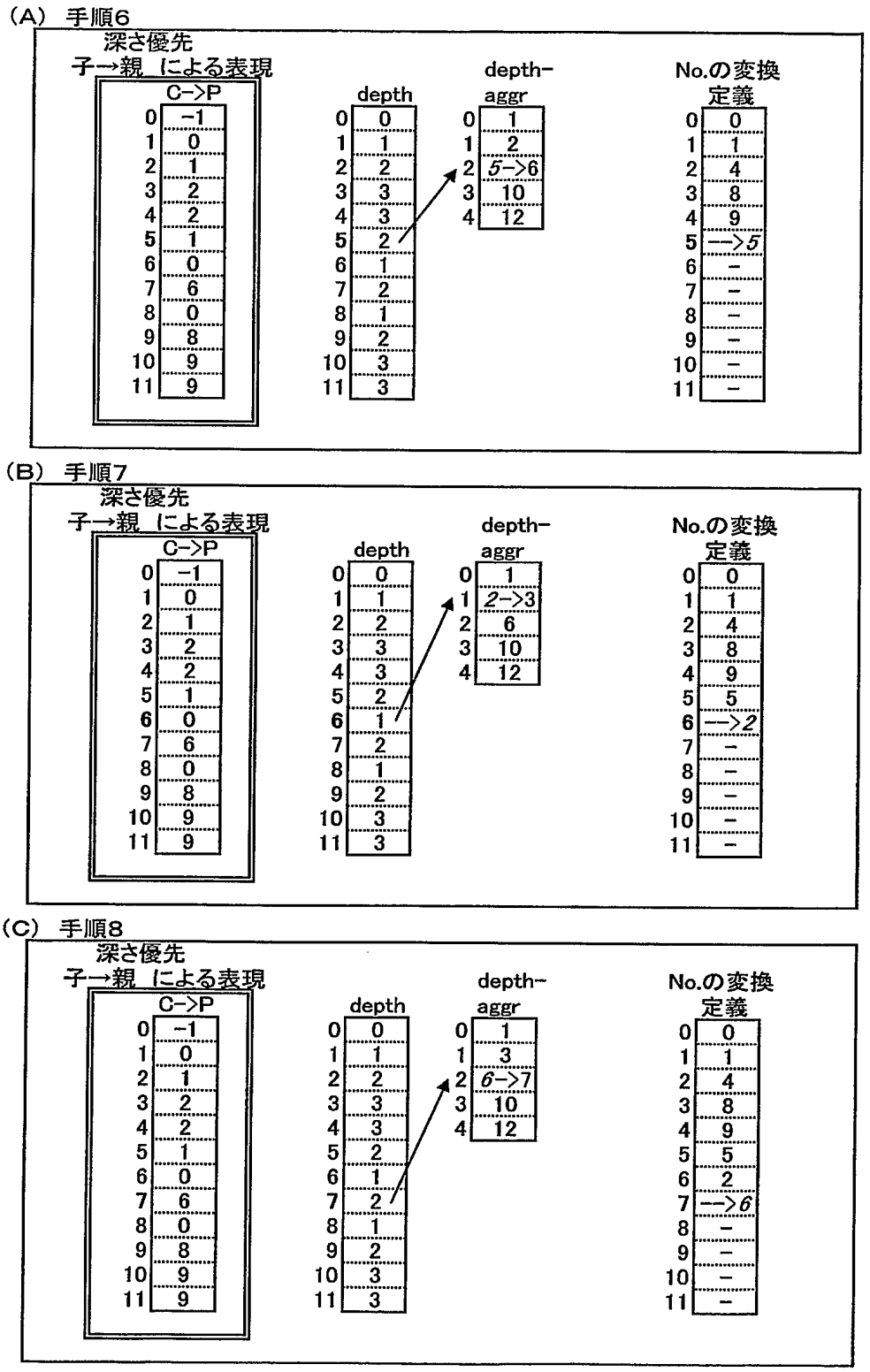
【図２５】

図25


(A) 手順3
深さ優先
子→親 による表現
C->P
0 -1
1 0
2 1
3 2
4 2
5 1
6 0
7 6
8 0
9 8
10 9
11 9
depth
0 0
1 1
2 2
3 3
4 3
5 2
6 1
7 2
8 1
9 2
10 3
11 3
depth-aggr
0 1
1 2
2 4->5
3 8
4 12
No.の変換定義
0 0
1 1
2 —>4
3 –
4 –
5 –
6 –
7 –
8 –
9 –
10 –
11 –
(B) 手順4
深さ優先
子→親 による表現
C->P
0 -1
1 0
2 1
3 2
4 2
5 1
6 0
7 6
8 0
9 8
10 9
11 9
depth
0 0
1 1
2 2
3 3
4 3
5 2
6 1
7 2
8 1
9 2
10 3
11 3
depth-aggr
0 1
1 2
2 5
3 8->9
4 12
No.の変換定義
0 0
1 1
2 4
3 —>8
4 –
5 –
6 –
7 –
8 –
9 –
10 –
11 –
(C) 手順5
深さ優先
子→親 による表現
C->P
0 -1
1 0
2 1
3 2
4 2
5 1
6 0
7 6
8 0
9 8
10 9
11 9
depth
0 0
1 1
2 2
3 3
4 3
5 2
6 1
7 2
8 1
9 2
10 3
11 3
depth-aggr
0 1
1 2
2 5
3 9->10
4 12
No.の変換定義
0 0
1 1
2 4
3 8
4 —>9
5 –
6 –
7 –
8 –
9 –
10 –
11 –

【図２６】

図26


(A) 手順6
深さ優先
子→親 による表現
C->P
depth
depth-aggr
No.の変換定義
(B) 手順7
深さ優先
子→親 による表現
C->P
depth
depth-aggr
No.の変換定義
(C) 手順8
深さ優先
子→親 による表現
C->P
depth
depth-aggr
No.の変換定義

【図２７】

図27

(A)　手順9

深さ優先
子→親　による表現

	C->P
0	-1
1	0
2	1
3	2
4	2
5	1
6	0
7	6
8	0
9	8
10	9
11	9

	depth
0	0
1	1
2	2
3	3
4	3
5	2
6	1
7	2
8	1
9	2
10	3
11	3

	depth-aggr
0	1
1	*3->4*
2	7
3	10
4	12

	No.の変換定義
0	0
1	1
2	4
3	8
4	9
5	5
6	2
7	6
8	*-->3*
9	-
10	-
11	-

(B)　手順10

深さ優先
子→親　による表現

	C->P
0	-1
1	0
2	1
3	2
4	2
5	1
6	0
7	6
8	0
9	8
10	9
11	9

	depth
0	0
1	1
2	2
3	3
4	3
5	2
6	1
7	2
8	1
9	2
10	3
11	3

	depth-aggr
0	1
1	4
2	*7->8*
3	10
4	12

	No.の変換定義
0	0
1	1
2	4
3	8
4	9
5	5
6	2
7	6
8	3
9	*-->7*
10	-
11	-

(C)　手順11

深さ優先
子→親　による表現

	C->P
0	-1
1	0
2	1
3	2
4	2
5	1
6	0
7	6
8	0
9	8
10	9
11	9

	depth
0	0
1	1
2	2
3	3
4	3
5	2
6	1
7	2
8	1
9	2
10	3
11	3

	depth-aggr
0	1
1	4
2	8
3	*10->11*
4	12

	No.の変換定義
0	0
1	1
2	4
3	8
4	9
5	5
6	2
7	6
8	3
9	7
10	*->10*
11	-

【図２８】

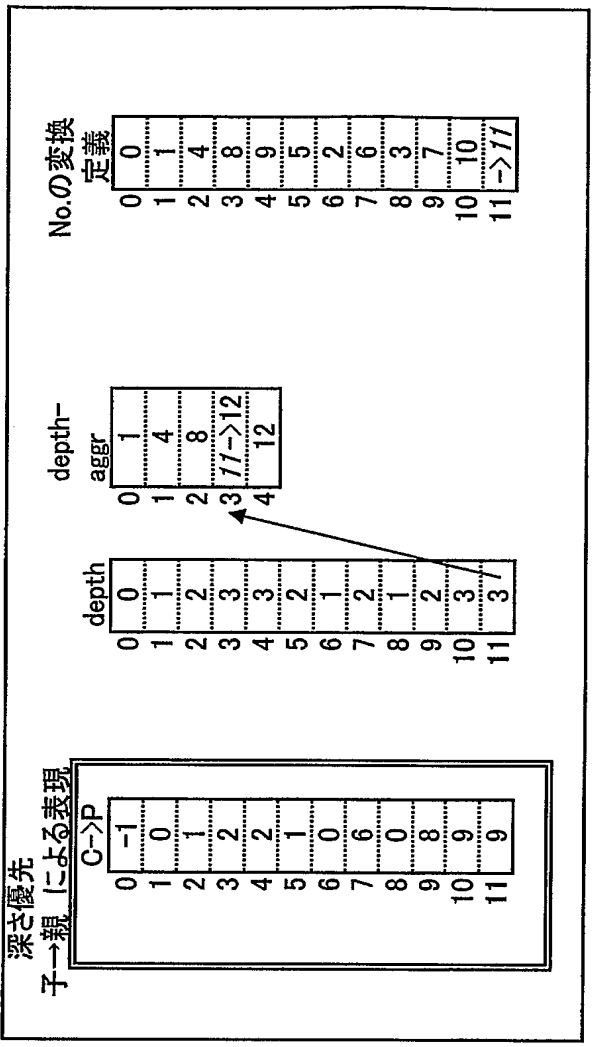


図28

【図２９】

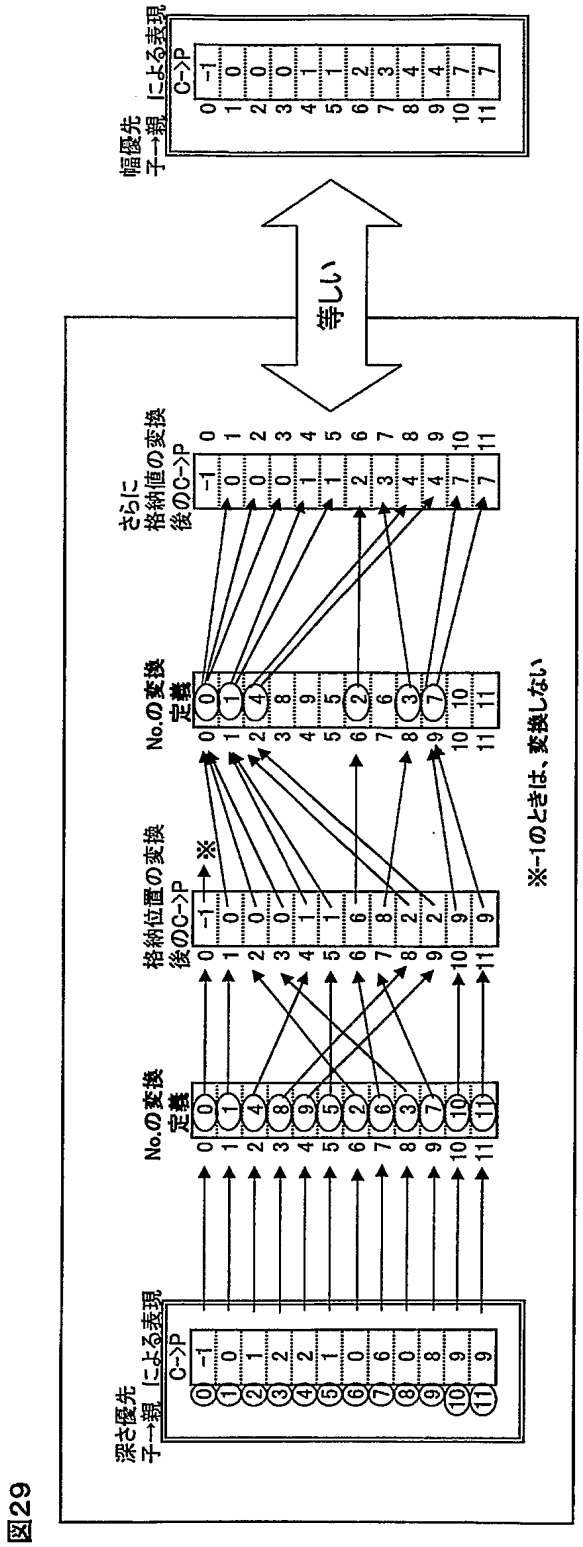

幅優先
子→親　による表現
C->P
等しい
さらに
格納値の変換
後のC->P
No.の変換
定義
格納位置の変換
後のC->P
※
※-1のときは、変換しない
深さ優先
子→親　による表現
C->P

図29

【図３０】

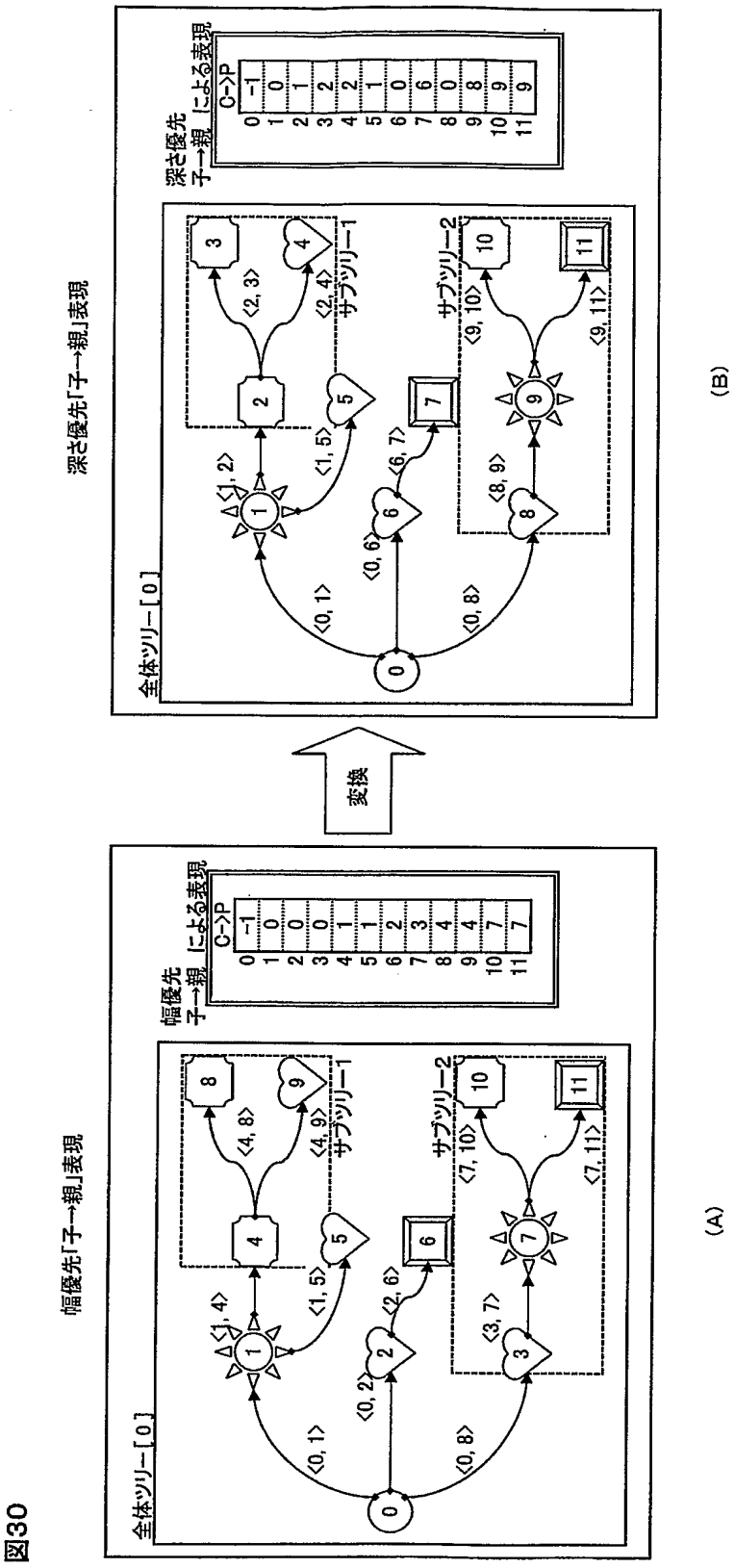
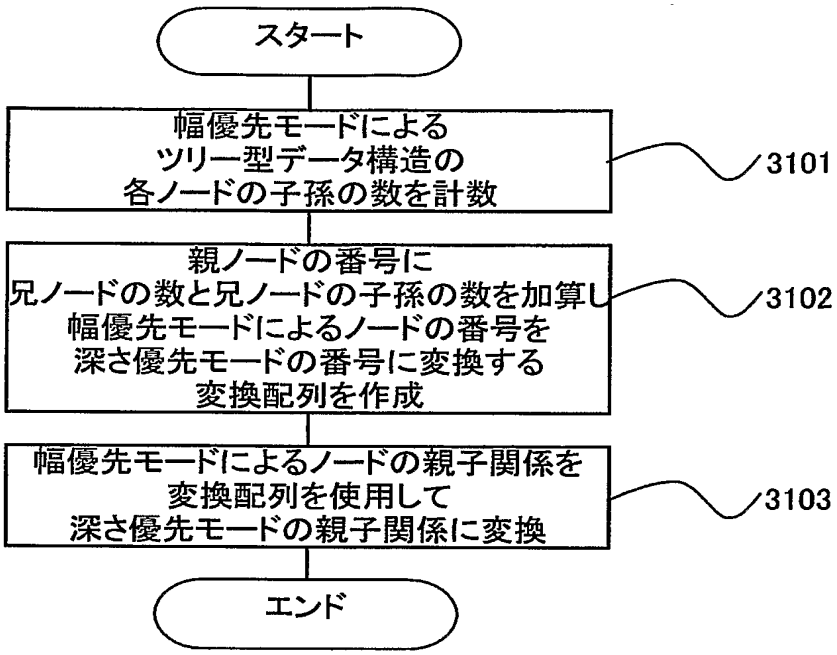


図30
幅優先「子→親」表現
全体ツリー[0]
幅優先 子→親 による表現
C->P
サブツリー1
サブツリー2
(A)
変換
深さ優先「子→親」表現
全体ツリー[0]
深さ優先 子→親 による表現
C->P
サブツリー1
サブツリー2
(B)

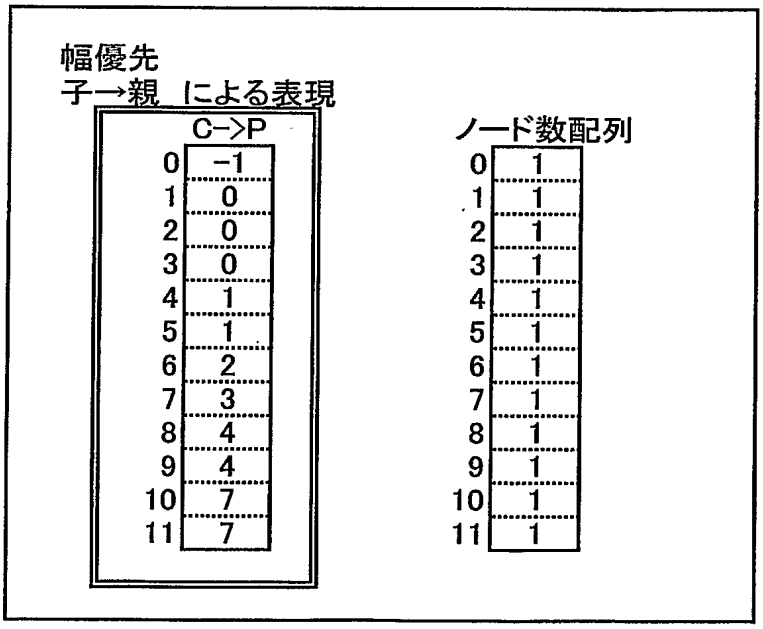
【図３１】

図31



【図３２】

図32



【図３３】

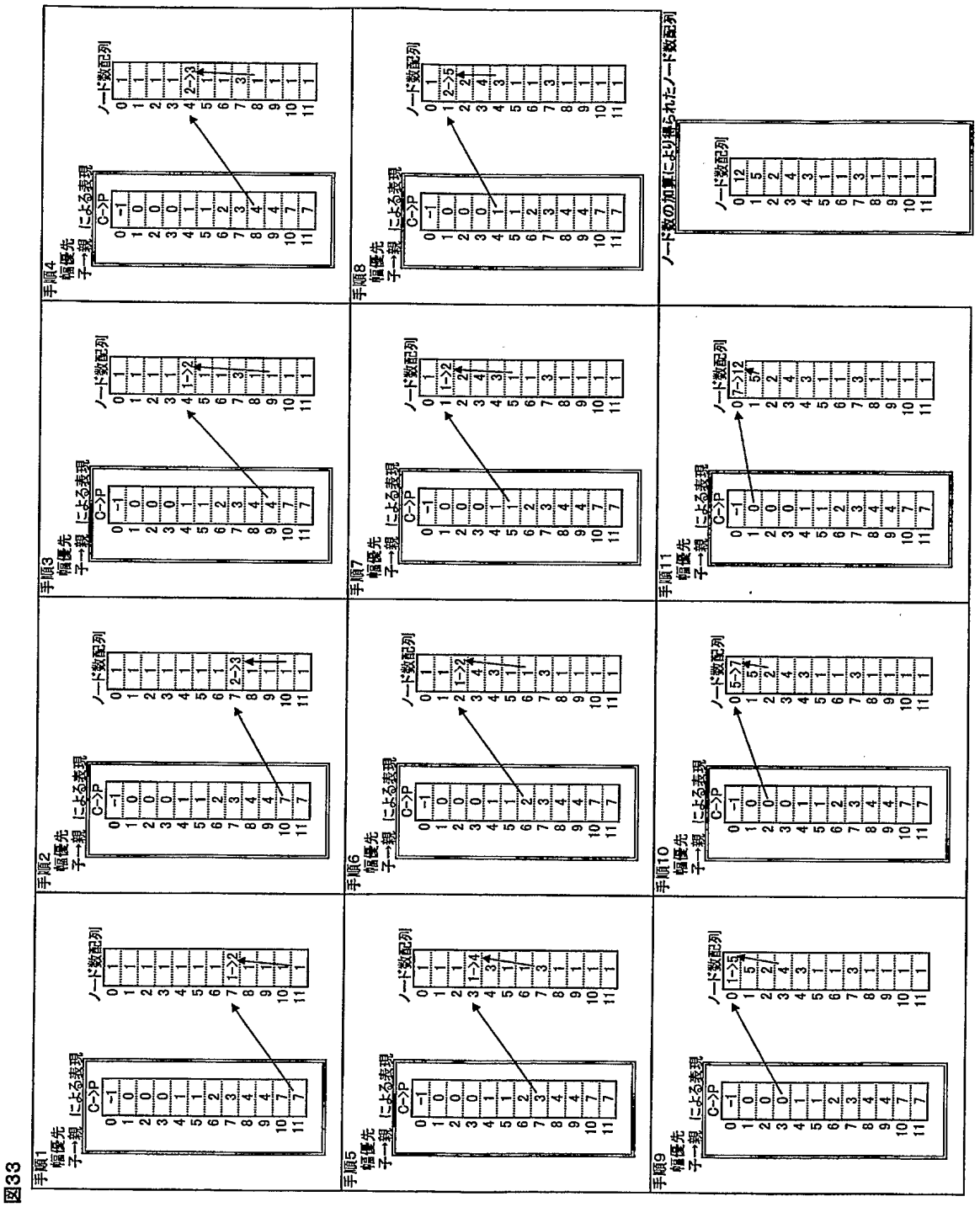


図33
手順1
幅優先
子→親　による表現
C->P
ノード数配列
手順2
手順3
手順4
手順5
手順6
手順7
手順8
手順9
手順10
手順11
ノード数の加算により構成されたノード数配列

【図３４】

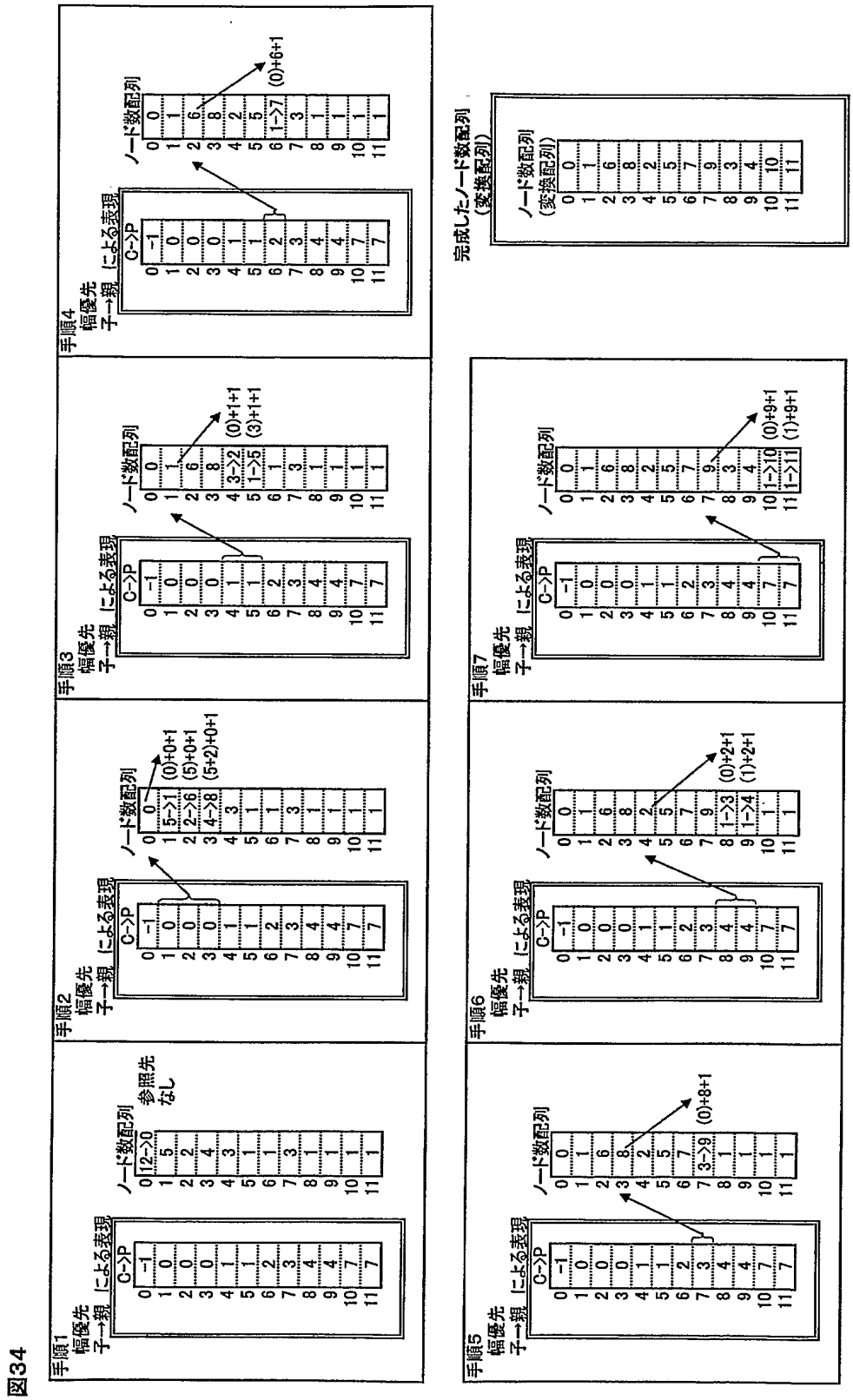

手順1
幅優先
子→親　による表現
C->P
ノード数配列
12->0
参照先なし
手順2
幅優先
子→親　による表現
C->P
ノード数配列
5->1
2->6
4->8
(0)+0+1
(5)+0+1
(5+2)+0+1
手順3
幅優先
子→親　による表現
C->P
ノード数配列
3->2
1->5
(0)+1+1
(3)+1+1
手順4
幅優先
子→親　による表現
C->P
ノード数配列
1->7
(0)+6+1
手順5
幅優先
子→親　による表現
C->P
ノード数配列
3->9
(0)+8+1
手順6
幅優先
子→親　による表現
C->P
ノード数配列
1->3
1->4
(0)+2+1
(1)+2+1
手順7
幅優先
子→親　による表現
C->P
ノード数配列
1->10
1->11
(0)+9+1
(1)+9+1
完成したノード数配列
(変換配列)
ノード数配列
(変換配列)

図34

【図３５】

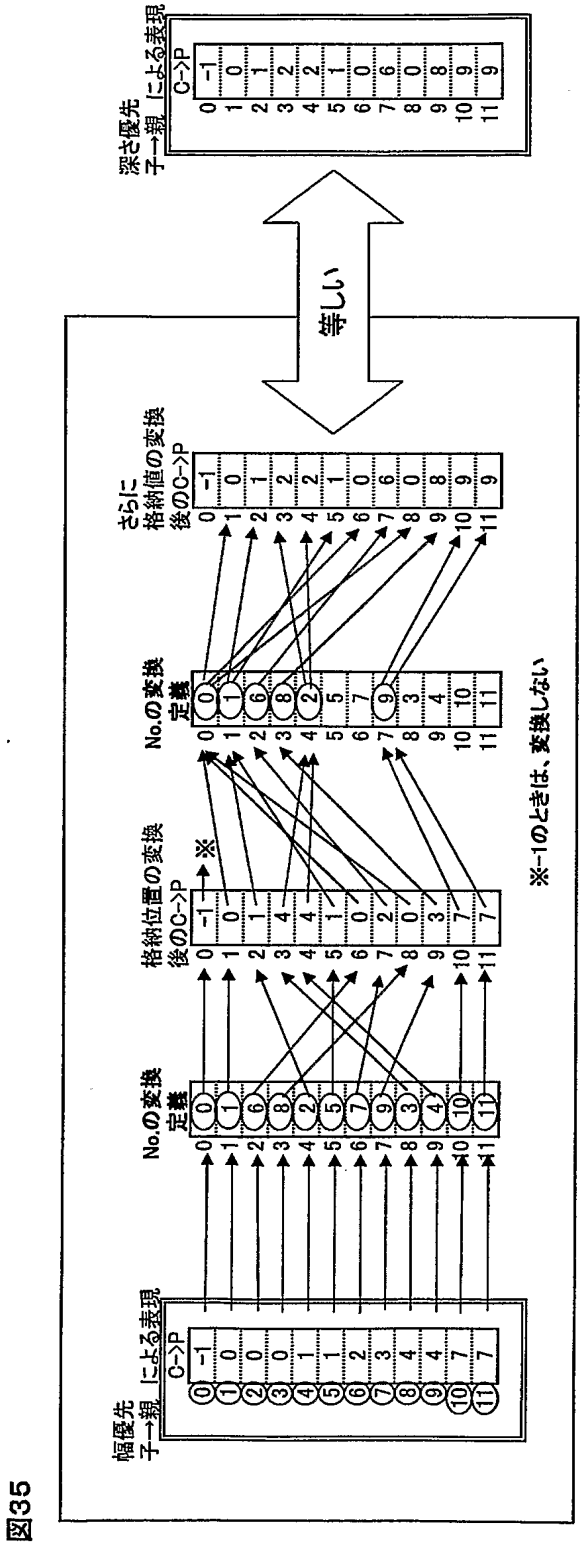
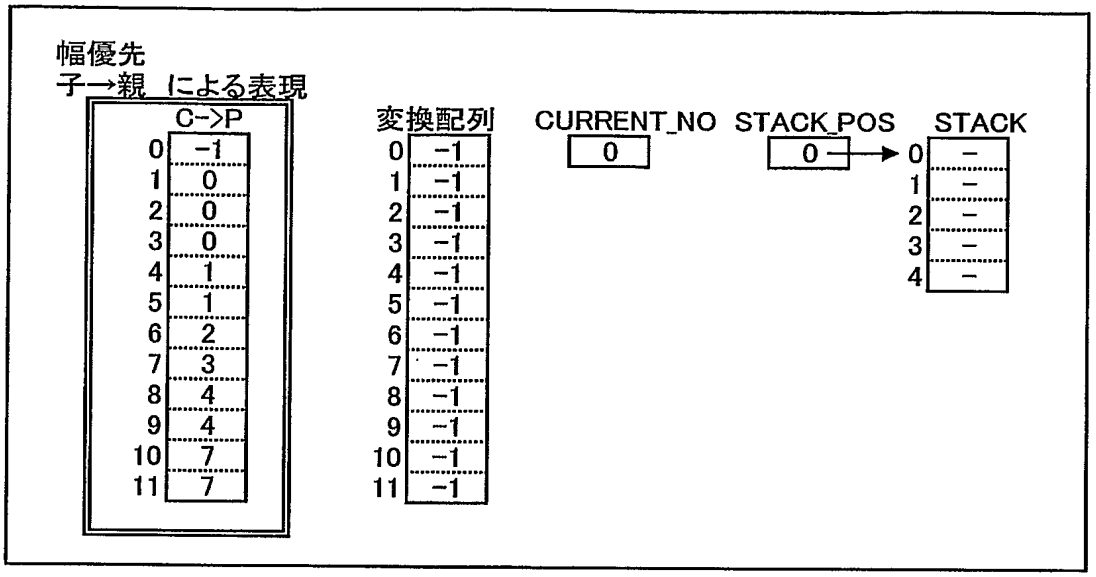

深さ優先
子→親　による表現
C->P
等しい
幅優先
子→親　による表現
C->P
さらに
格納値の変換
後のC->P
No.の変換
定義
格納位置の変換
後のC->P
※
※-1のときは、変換しない

図35

【図３６】

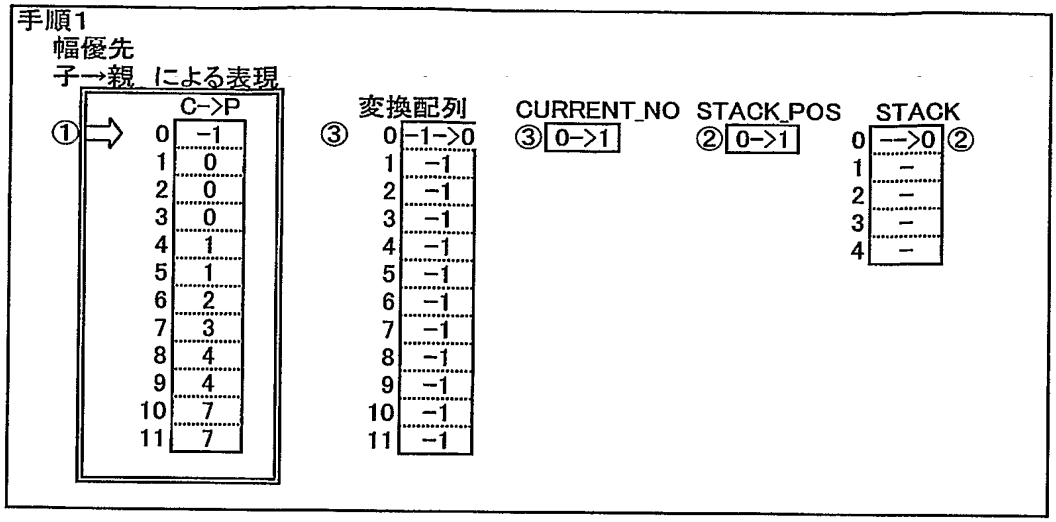
図36


幅優先
子→親　による表現
C->P
0 -1
1 0
2 0
3 0
4 1
5 1
6 2
7 3
8 4
9 4
10 7
11 7
変換配列
0 -1
1 -1
2 -1
3 -1
4 -1
5 -1
6 -1
7 -1
8 -1
9 -1
10 -1
11 -1
CURRENT_NO
0
STACK_POS
0
STACK
0 –
1 –
2 –
3 –
4 –

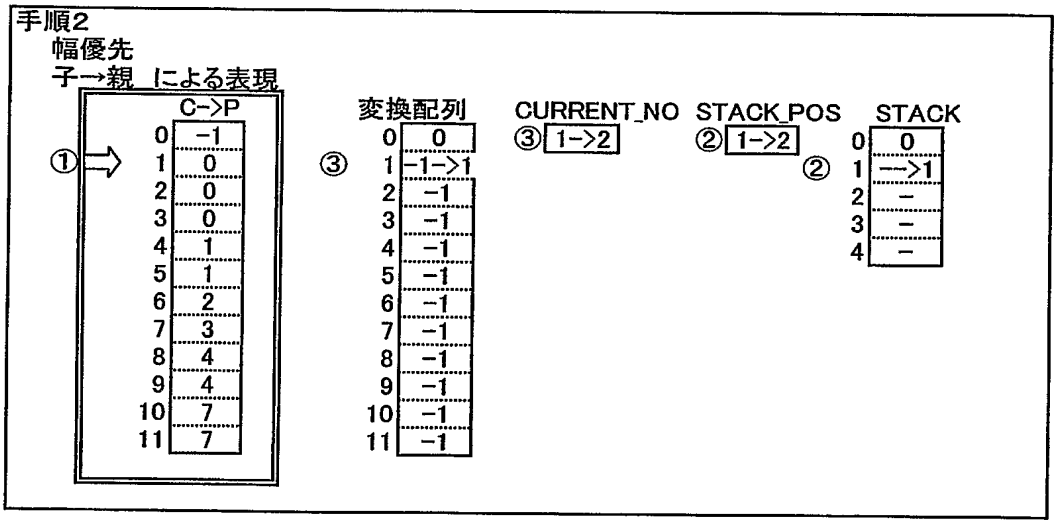
【図３７】

図37

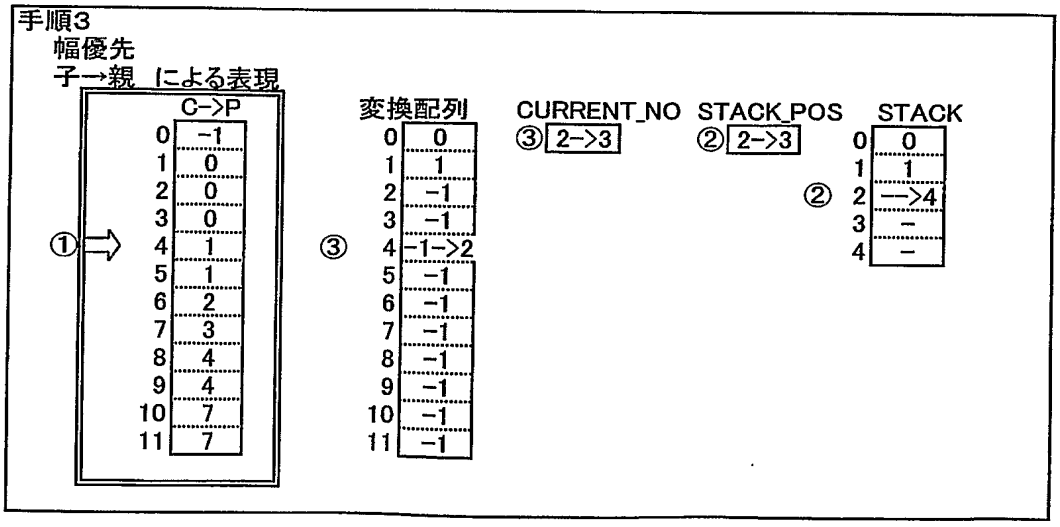
(A)


手順1
幅優先
子→親　による表現
C->P
①
0 -1
1 0
2 0
3 0
4 1
5 1
6 2
7 3
8 4
9 4
10 7
11 7
③
変換配列
0 -1->0
1 -1
2 -1
3 -1
4 -1
5 -1
6 -1
7 -1
8 -1
9 -1
10 -1
11 -1
CURRENT_NO
③ 0->1
STACK_POS
② 0->1
STACK
0 -->0 ②
1 –
2 –
3 –
4 –

(B)


手順2
幅優先
子→親　による表現
C->P
①
0 -1
1 0
2 0
3 0
4 1
5 1
6 2
7 3
8 4
9 4
10 7
11 7
③
変換配列
0 0
1 -1->1
2 -1
3 -1
4 -1
5 -1
6 -1
7 -1
8 -1
9 -1
10 -1
11 -1
CURRENT_NO
③ 1->2
STACK_POS
② 1->2
STACK
0 0
1 -->1
② 2 –
3 –
4 –

(C)


手順3
幅優先
子→親　による表現
C->P
①
0 -1
1 0
2 0
3 0
4 1
5 1
6 2
7 3
8 4
9 4
10 7
11 7
③
変換配列
0 0
1 1
2 -1
3 -1
4 -1->2
5 -1
6 -1
7 -1
8 -1
9 -1
10 -1
11 -1
CURRENT_NO
③ 2->3
STACK_POS
② 2->3
STACK
0 0
1 1
② 2 -->4
3 –
4 –

【図３８】

図38

(A)

手順4
幅優先
子→親　による表現

	C->P
0	-1
1	0
2	0
3	0
4	1
5	1
6	2
7	3
① ⇒ 8	4
9	4
10	7
11	7

変換配列

0	0
1	1
2	-1
3	-1
4	2
5	-1
6	-1
7	-1
③ 8	-1->3
9	-1
10	-1
11	-1

CURRENT_NO
③ 3->4

STACK_POS
② 3->4

STACK

0	0
1	1
2	4
② 3	-->8
4	-

(B)

手順5
幅優先
子→親　による表現

	C->P
0	-1
1	0
2	0
3	0
4	1
5	1
6	2
7	3
8	4
9	4
10	7
11	7

変換配列

0	0
1	1
2	-1
3	-1
4	2
5	-1
6	-1
7	-1
8	3
9	-1
10	-1
11	-1

CURRENT_NO
4

STACK_POS
① 4->3

STACK

0	0
1	1
2	4
① 3	~~3~~
4	-

(C)

手順6
幅優先
子→親　による表現

	C->P
0	-1
1	0
2	0
3	0
4	1
5	1
6	2
7	3
8	4
① ⇒ 9	4
10	7
11	7

変換配列

0	0
1	1
2	-1
3	-1
4	2
5	-1
6	-1
7	-1
8	3
③ 9	-1->4
10	-1
11	-1

CURRENT_NO
③ 4->5

STACK_POS
② 3->4

STACK

0	0
1	1
2	4
② 3	-->9
4	-

【図３９】

図39

(A)

手順7
幅優先
子→親　による表現

	C->P
0	-1
1	0
2	0
3	0
4	1
5	1
6	2
7	3
8	4
9	4
10	7
11	7

変換配列

0	0
1	1
2	-1
3	-1
4	2
5	-1
6	-1
7	-1
8	3
9	4
10	-1
11	-1

CURRENT_NO: 5

STACK_POS: ① 4->3

STACK

0	0
1	1
2	4
① 3	~~0~~
4	-

(B)

手順8
幅優先
子→親　による表現

	C->P
0	-1
1	0
2	0
3	0
4	1
5	1
6	2
7	3
8	4
9	4
10	7
11	7

変換配列

0	0
1	1
2	-1
3	-1
4	2
5	-1
6	-1
7	-1
8	3
9	4
10	-1
11	-1

CURRENT_NO: 5

STACK_POS: ① 3->2

STACK

0	0
1	1
① 2	~~4~~
3	-
4	-

(C)

手順9
幅優先
子→親　による表現

	C->P
0	-1
1	0
2	0
3	0
4	1
① 5	1
6	2
7	3
8	4
9	4
10	7
11	7

③

変換配列

0	0
1	1
2	-1
3	-1
4	2
5	-1->5
6	-1
7	-1
8	3
9	4
10	-1
11	-1

CURRENT_NO: ③ 5->6

STACK_POS: ② 2->3

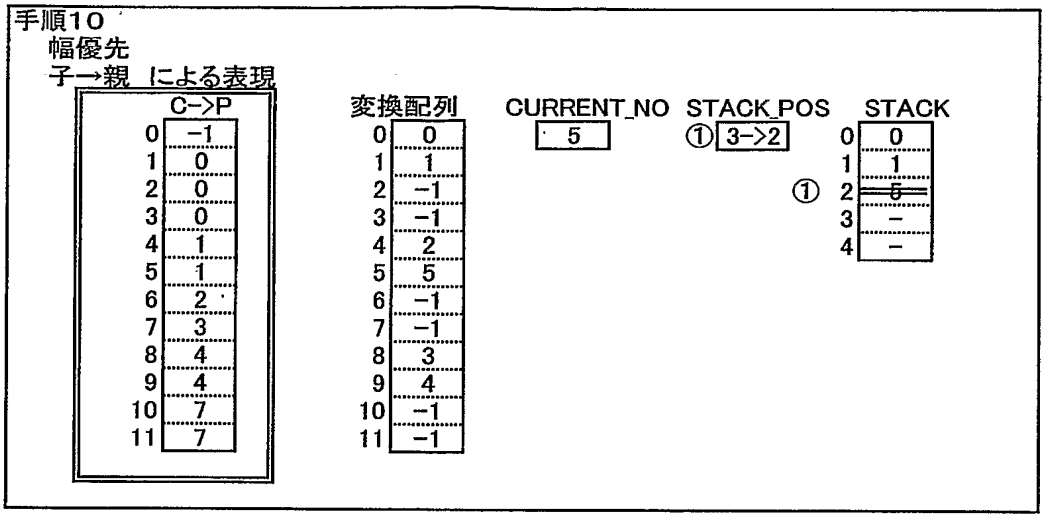
STACK

0	0
1	1
② 2	-->5
3	-
4	-

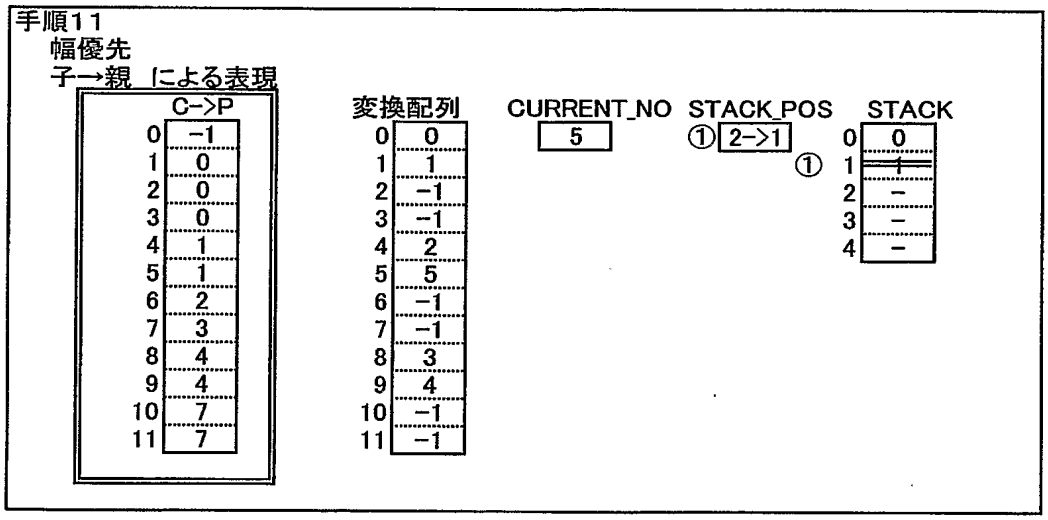
【図４０】

図40

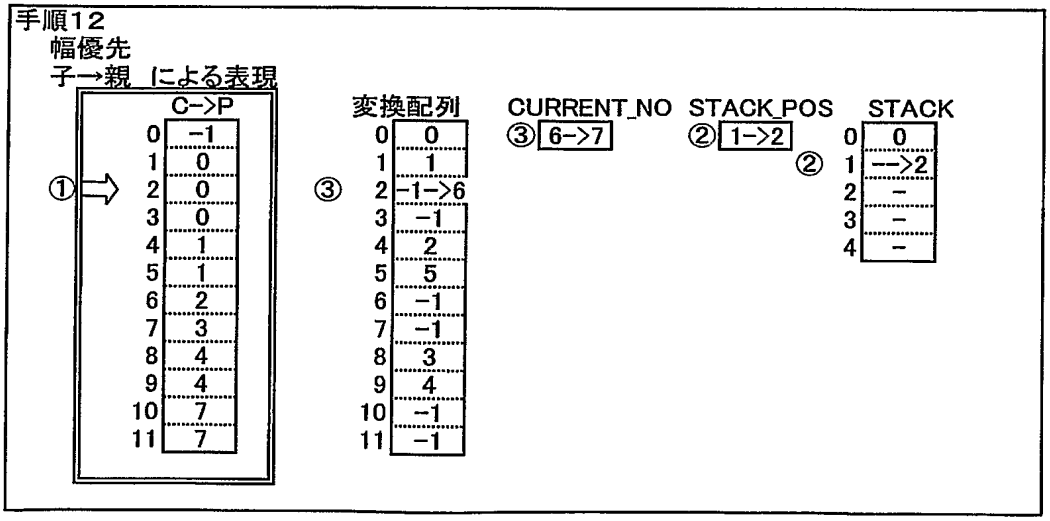
(A)


手順10
幅優先
子→親　による表現
C->P
0 −1
1 0
2 0
3 0
4 1
5 1
6 2
7 3
8 4
9 4
10 7
11 7
変換配列
0 0
1 1
2 −1
3 −1
4 2
5 5
6 −1
7 −1
8 3
9 4
10 −1
11 −1
CURRENT_NO
5
STACK_POS
①3->2
STACK
0 0
1 1
① 2 5
3 −
4 −

(B)

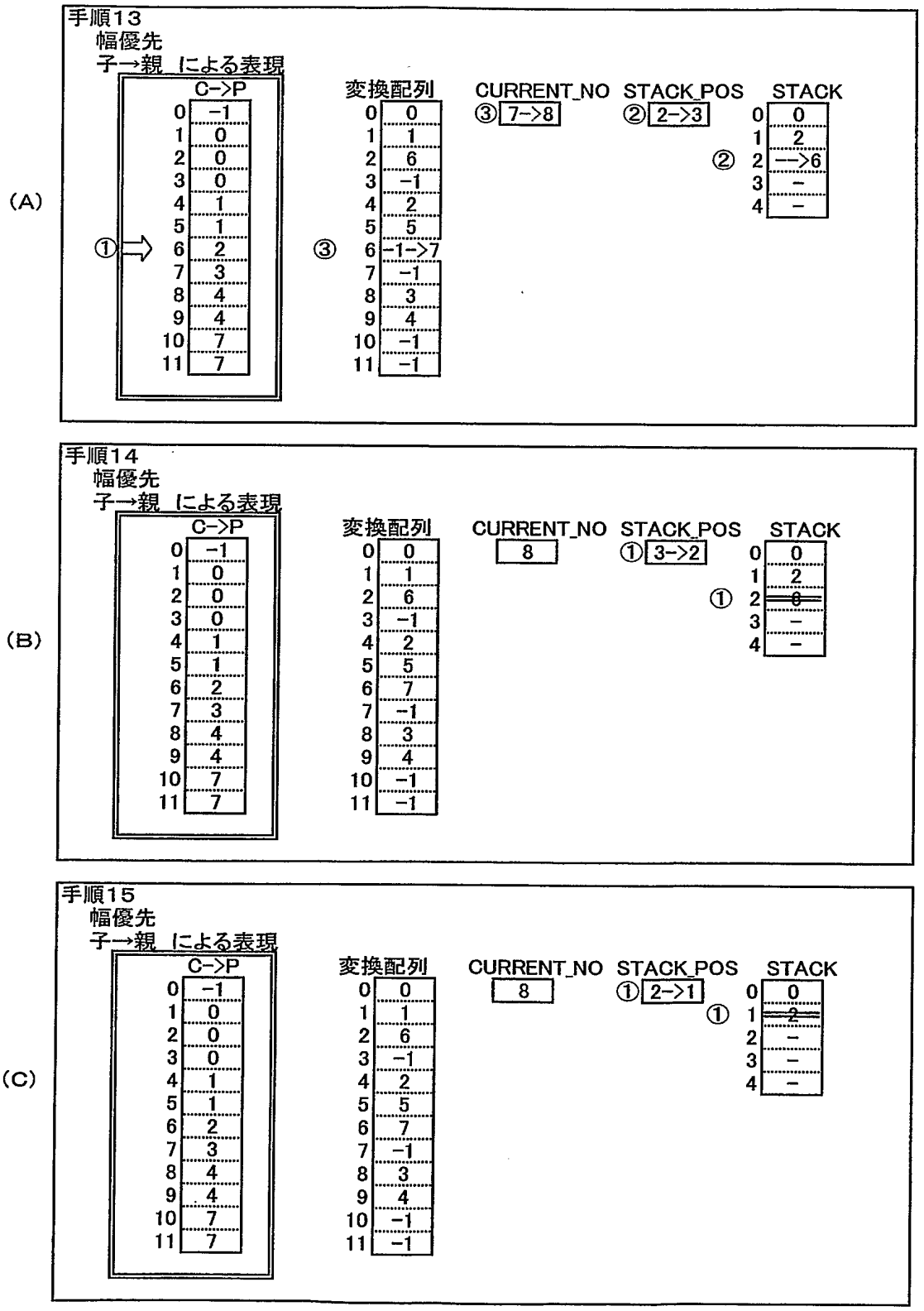

手順11
幅優先
子→親　による表現
C->P
0 −1
1 0
2 0
3 0
4 1
5 1
6 2
7 3
8 4
9 4
10 7
11 7
変換配列
0 0
1 1
2 −1
3 −1
4 2
5 5
6 −1
7 −1
8 3
9 4
10 −1
11 −1
CURRENT_NO
5
STACK_POS
①2->1
STACK
0 0
① 1 1
2 −
3 −
4 −

(C)


手順12
幅優先
子→親　による表現
C->P
① 0 −1
1 0
2 0
3 0
4 1
5 1
6 2
7 3
8 4
9 4
10 7
11 7
③
変換配列
0 0
1 1
2 −1->6
3 −1
4 2
5 5
6 −1
7 −1
8 3
9 4
10 −1
11 −1
CURRENT_NO
③6->7
STACK_POS
②1->2
STACK
0 0
② 1 −−>2
2 −
3 −
4 −

【図４１】

図41


手順13
幅優先
子→親 による表現
(A)
C->P
0 -1
1 0
2 0
3 0
4 1
5 1
① 6 2
7 3
8 4
9 4
10 7
11 7
変換配列
③
0 0
1 1
2 6
3 -1
4 2
5 5
6 -1->7
7 -1
8 3
9 4
10 -1
11 -1
CURRENT_NO
③ 7->8
STACK_POS
② 2->3
STACK
0 0
1 2
② 2 —>6
3 –
4 –
手順14
幅優先
子→親 による表現
(B)
C->P
0 -1
1 0
2 0
3 0
4 1
5 1
6 2
7 3
8 4
9 4
10 7
11 7
変換配列
0 0
1 1
2 6
3 -1
4 2
5 5
6 7
7 -1
8 3
9 4
10 -1
11 -1
CURRENT_NO
8
STACK_POS
① 3->2
STACK
0 0
1 2
① 2 6
3 –
4 –
手順15
幅優先
子→親 による表現
(C)
C->P
0 -1
1 0
2 0
3 0
4 1
5 1
6 2
7 3
8 4
9 4
10 7
11 7
変換配列
0 0
1 1
2 6
3 -1
4 2
5 5
6 7
7 -1
8 3
9 4
10 -1
11 -1
CURRENT_NO
8
STACK_POS
① 2->1
STACK
0 0
① 1 2
2 –
3 –
4 –

【図４２】

図42

(A)

手順16
兄弟優先
子→親　による表現

	C->P
0	-1
1	0
2	0
① 3	0
4	1
5	1
6	2
7	3
8	4
9	4
10	7
11	7

変換配列

0	0
1	1
2	6
③ 3	-1->8
4	2
5	5
6	7
7	-1
8	3
9	4
10	-1
11	-1

CURRENT_NO
③ 8->9

STACK_POS
② 1->2

STACK

0	0
② 1	-->3
2	-
3	-
4	-

(B)

手順17
兄弟優先
子→親　による表現

	C->P
0	-1
1	0
2	0
3	0
4	1
5	1
6	2
① 7	3
8	4
9	4
10	7
11	7

変換配列

0	0
1	1
2	6
3	8
4	2
5	5
6	7
③ 7	-1->9
8	3
9	4
10	-1
11	-1

CURRENT_NO
③ 9->10

STACK_POS
② 2->3

STACK

0	0
1	3
② 2	-->7
3	-
4	-

(C)

手順18
兄弟優先
子→親　による表現

	C->P
0	-1
1	0
2	0
3	0
4	1
5	1
6	2
7	3
8	4
9	4
① 10	7
11	7

変換配列

0	0
1	1
2	6
3	8
4	2
5	5
6	7
7	8
8	3
9	4
③ 10	-1->10
11	-1

CURRENT_NO
③ 10->11

STACK_POS
② 3->4

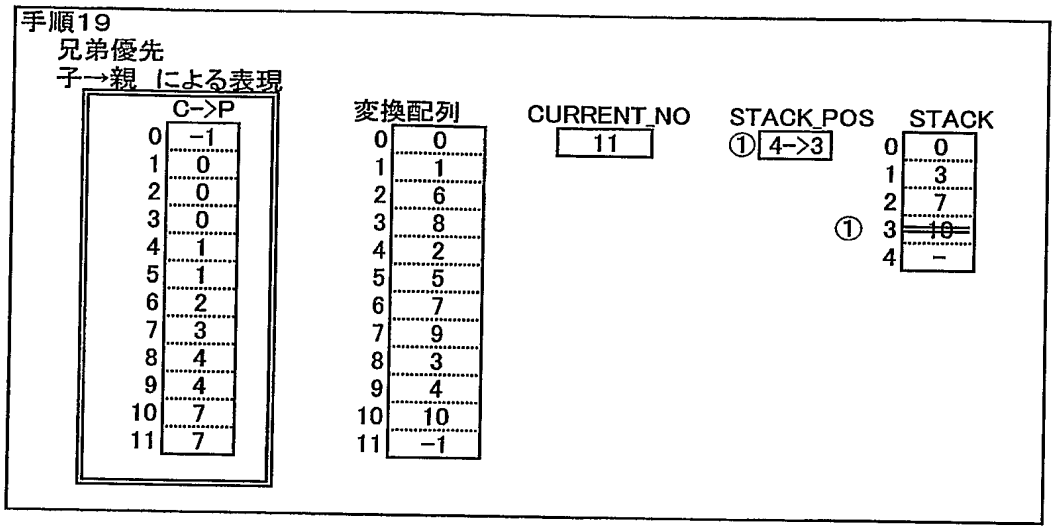
STACK

0	0
1	3
2	7
② 3	-->10
4	-

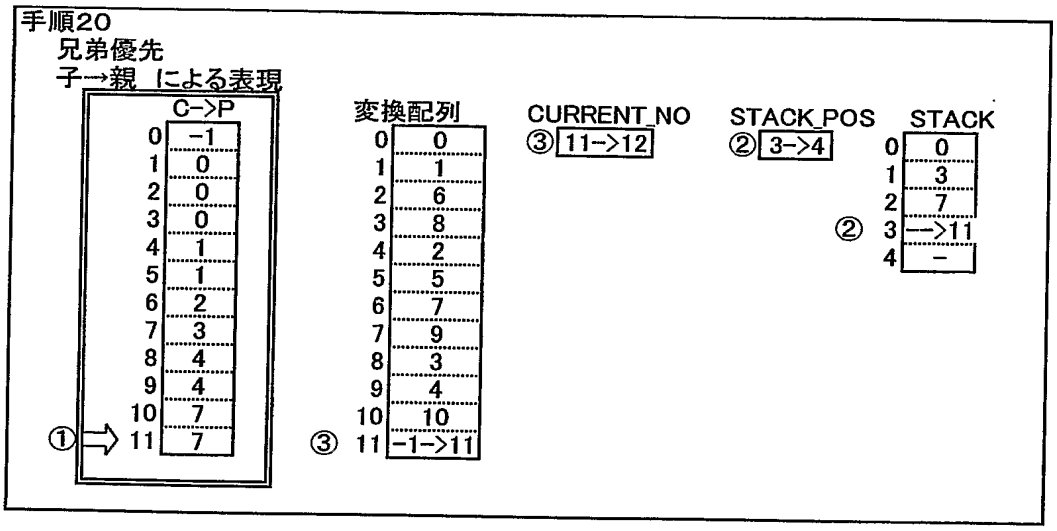
【図４３】

図43

(A)

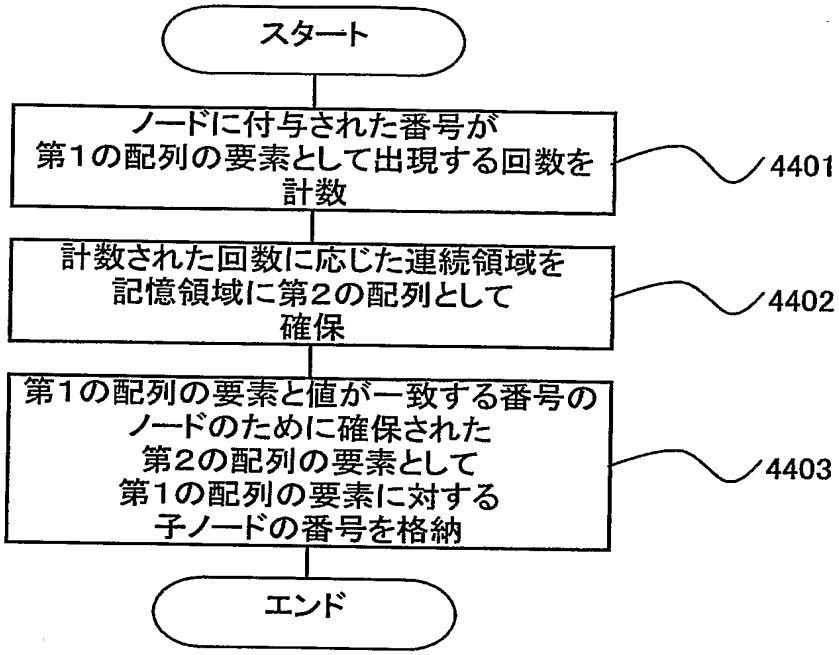

手順19
兄弟優先
子→親　による表現
C->P
0 -1
1 0
2 0
3 0
4 1
5 1
6 2
7 3
8 4
9 4
10 7
11 7
変換配列
0 0
1 1
2 6
3 8
4 2
5 5
6 7
7 9
8 3
9 4
10 10
11 -1
CURRENT_NO
11
STACK_POS
① 4->3
STACK
0 0
1 3
2 7
① 3 ~~10~~
4 -

(B)


手順20
兄弟優先
子→親　による表現
C->P
0 -1
1 0
2 0
3 0
4 1
5 1
6 2
7 3
8 4
9 4
10 7
① 11 7
変換配列
0 0
1 1
2 6
3 8
4 2
5 5
6 7
7 9
8 3
9 4
10 10
③ 11 -1->11
CURRENT_NO
③ 11->12
STACK_POS
② 3->4
STACK
0 0
1 3
2 7
② 3 —>11
4 -

【図４４】

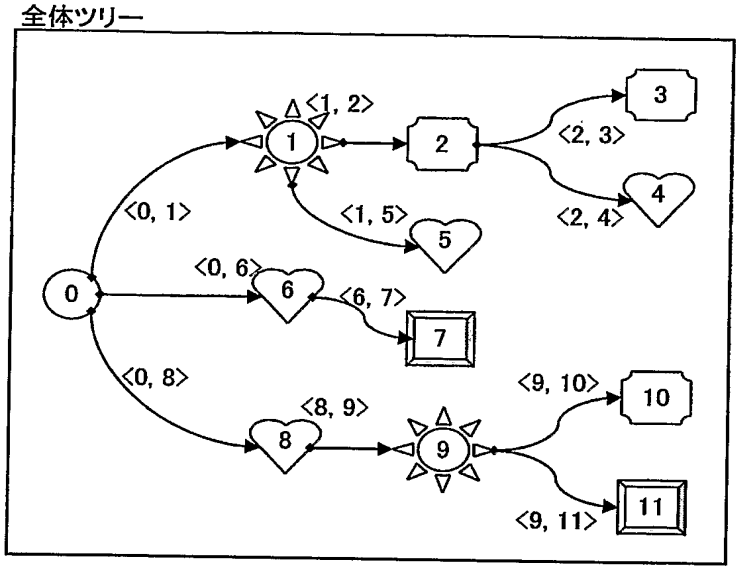
図44


スタート
ノードに付与された番号が
第1の配列の要素として出現する回数を
計数
4401
計数された回数に応じた連続領域を
記憶領域に第2の配列として
確保
4402
第1の配列の要素と値が一致する番号の
ノードのために確保された
第2の配列の要素として
第1の配列の要素に対する
子ノードの番号を格納
4403
エンド

【図４５】

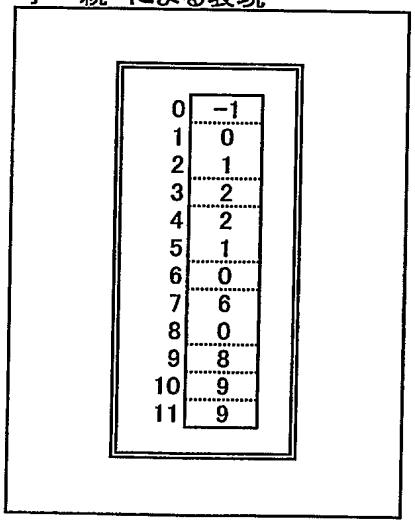
図45

全体ツリー



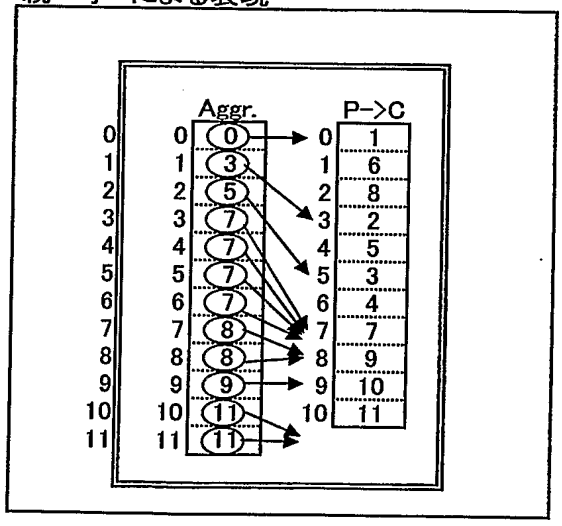
(A)

子→親　による表現



(B)

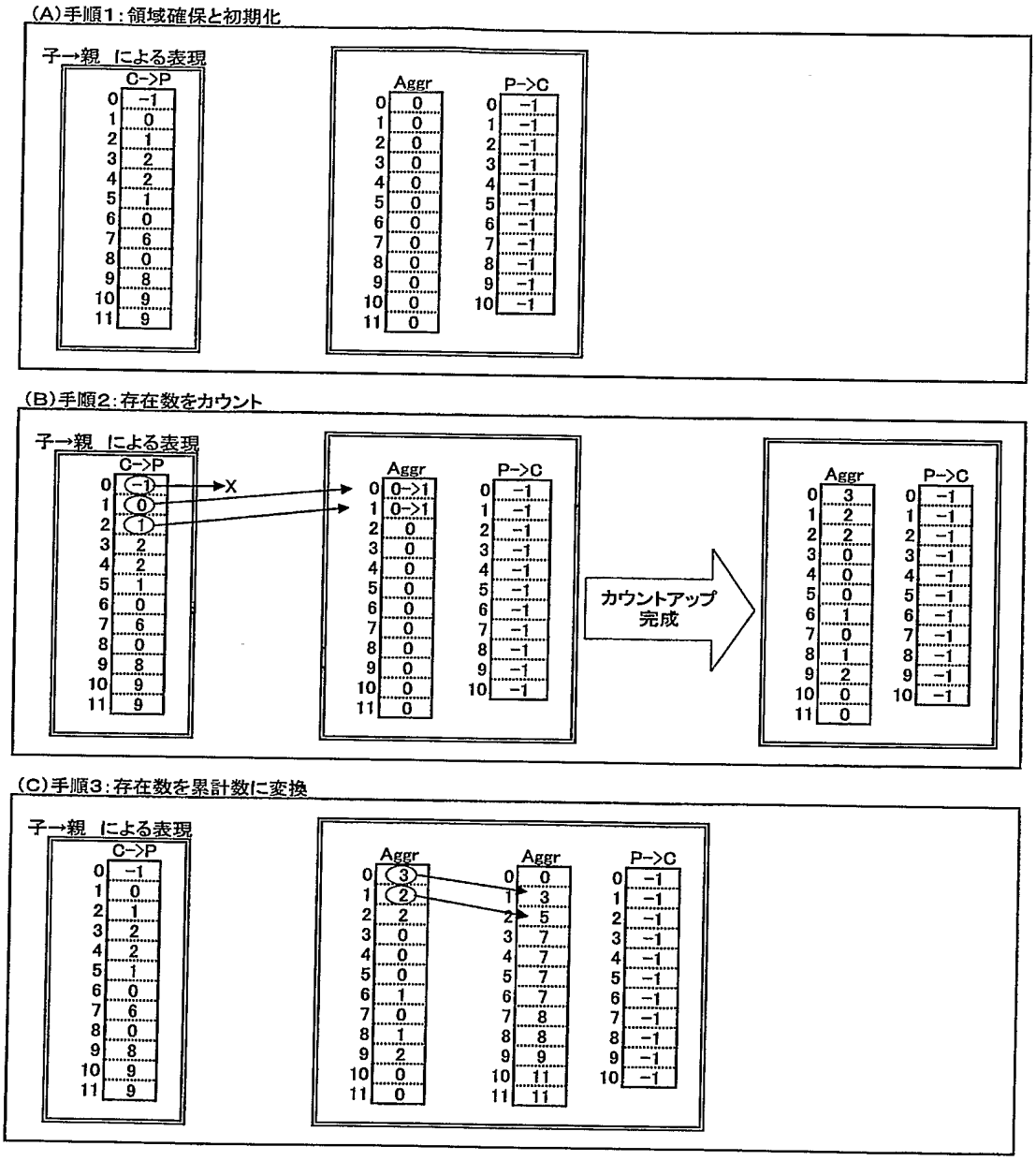
親→子　による表現



(C)

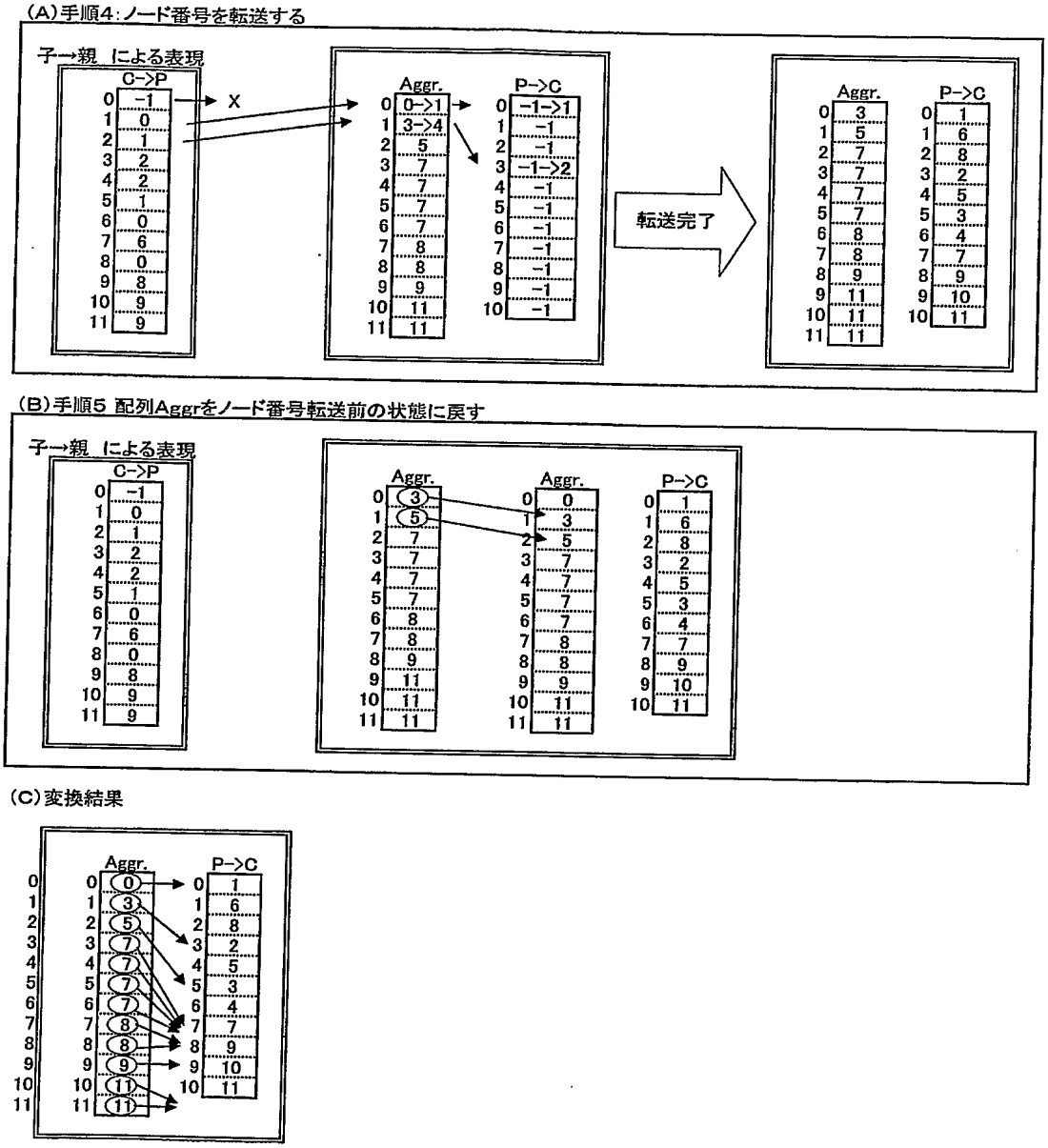
【図４６】

図46


(A)手順1:領域確保と初期化
子→親 による表現
C->P
Aggr
P->C
(B)手順2:存在数をカウント
X
カウントアップ
完成
(C)手順3:存在数を累計数に変換

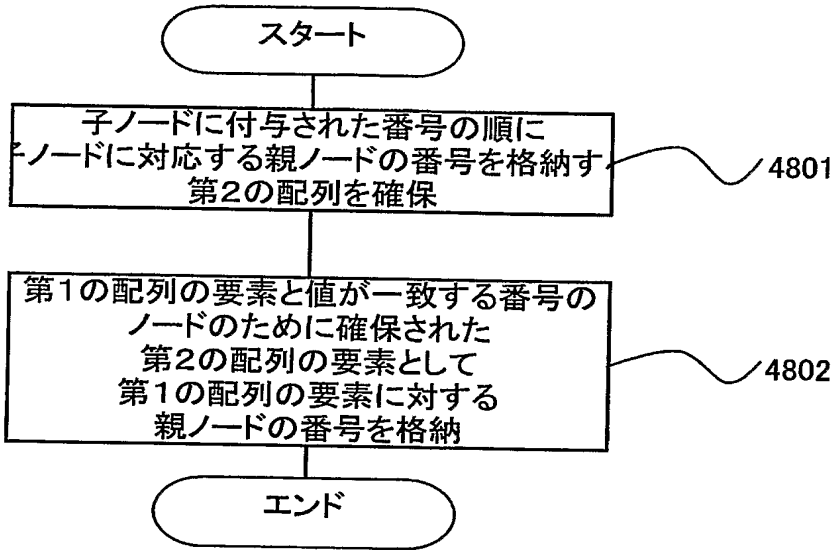
【図４７】

図47

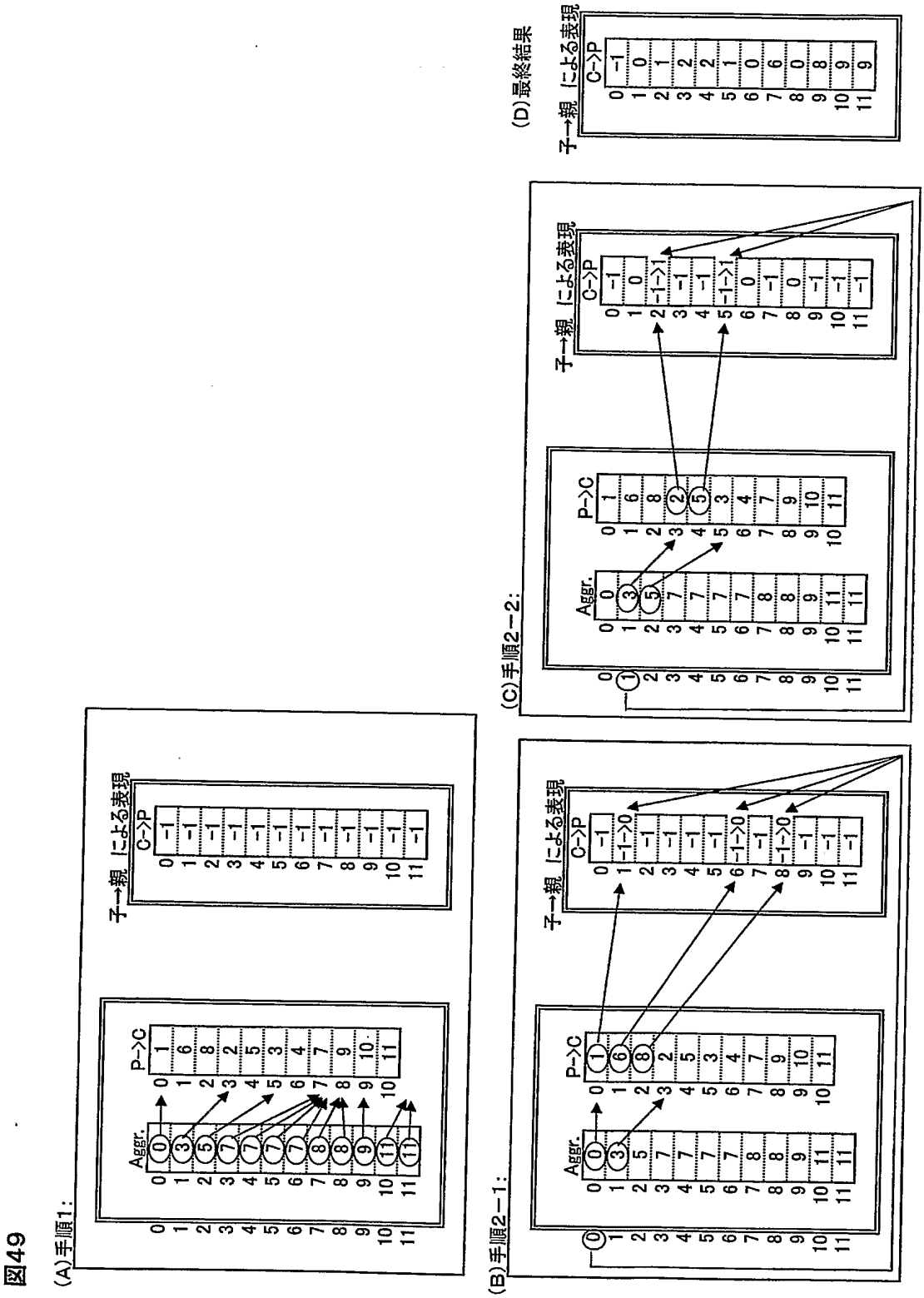

(A)手順4:ノード番号を転送する
子→親 による表現
C->P
Aggr.
P->C
X
0->1
3->4
-1->1
-1->2
転送完了
(B)手順5 配列Aggrをノード番号転送前の状態に戻す
子→親 による表現
C->P
Aggr.
Aggr.
P->C
(C)変換結果
Aggr.
P->C

【図４８】

図48

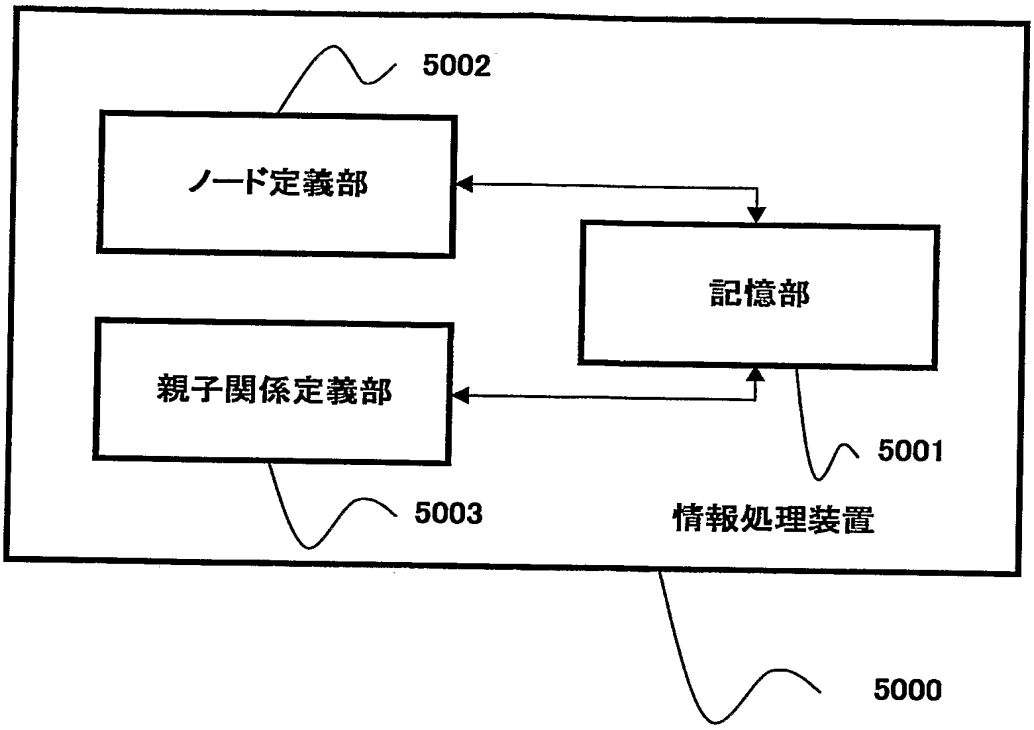

スタート
子ノードに付与された番号の順に
子ノードに対応する親ノードの番号を格納す
第2の配列を確保
4801
第1の配列の要素と値が一致する番号の
ノードのために確保された
第2の配列の要素として
第1の配列の要素に対する
親ノードの番号を格納
4802
エンド

【図４９】



【図５０】

図50



【書類名】要約書
【要約】
【課題】　ツリー型データ構造のデータ間の関係、例えば、親子、祖先、子孫、兄弟、世代などの関係を効率的にトレースすることができるツリー型データ構造の表現方法を提供する。
【解決手段】　ツリー型データ構造を構成するノード間の親子関係は、子ノードに親ノードを対応付ける「子→親」関係によって表現される。「子→親」関係によって親子関係を表現する場合、一つの子ノードには必ず唯一の親ノードが対応するので、子ノードを特定することによって、この子ノードに対応する唯一の親ノードを直ちに特定することができる。このため、本発明は、ルート・ノード以外のノードである非ルート・ノードの各々に対して、非ルート・ノードの親ノードを関連付けることにより前記ノード間の親子関係を表現する、
【選択図】　図１６

認定・付加情報

特許出願の番号	特願２００４－０７５４０４
受付番号	５０４００４３５３６８
書類名	特許願
担当官	第七担当上席　　００９６
作成日	平成１６年　３月２５日

＜認定情報・付加情報＞

【提出日】　平成16年 3月16日

特願２００４－０７５４０４

出　願　人　履　歴　情　報

識別番号　［５０２３６９０１２］

１．変更年月日　２００２年　９月　５日
［変更理由］　新規登録
住　所　神奈川県横浜市神奈川区松見町四丁目１１０１番地７
氏　名　株式会社ターボデータラボラトリー